technische universität
dortmund

department of
computer science

TECHNICAL REPORTS IN COMPUTER SCIENCE

Technische Universität Dortmund

Genetische Merkmalsselektion für die Gamma-Hadron-Separation im MAGIC-Experiment

Marius Helf, Katharina Morik

Informatik Lehrstuhl VIII, Künstliche Intelligenz

Wolfgang Rhode

Physik Lehrstuhl E5b, Astroteilchenphysik

Nummer: 828
November 2009

Technische Universität Dortmund — Fakultät für Informatik
Otto-Hahn-Str. 14, 44227 Dortmund

Marius Helf, Katharina Morik (Fakultät Informatik), Wolfgang Rhode (Fakultät Physik): *Genetische Merkmalsselektion für die Gamma-Hadron-Separation im MAGIC-Experiment,* Technical Report, Technische Universität Dortmund. © November 2009

INHALTSVERZEICHNIS

1 EINLEITUNG

1.1 ZIELE UND MOTIVATION

Das MAGIC-Teleskop[1] auf La Palma registriert Cherenkov-Licht von hochenergetischen kosmischen Teilchen. Seine Primäraufgabe besteht darin, Gamma-Teilchen zu beobachten, die von den Kernen Aktiver Galaxien (AGN) erzeugt und ausgesendet werden. Dazu ist eine gute Separation dieser Teilchen vom hadronischen Hintergrund notwendig. Diese Aufgabe übernimmt derzeit die speziell hierzu entwickelte Analysesoftware MARS [8], die auf die am CERN entwickelten ROOT-Bibliothek [21] aufbaut.

Die von den Teilchen erzeugten Lichtblitze, *Events*, werden von einer Kamera mit 578 Pixeln aufgenommen. MARS beinhaltet Programme zur Verarbeitung und Analyse dieser Kameradaten. Insbesondere werden aus den Rohdaten in mehreren Schritten Merkmale, hier *Parameter* genannt, extrahiert, anhand derer die Klassifikation vorgenommen wird. Eine zentrale Rolle spielen dabei die sogenannten Hillas-Parameter [15] und weitere Parameter der näherungsweise ellipsenförmigen Abbildungen.

Somit bietet MARS eine Oberfläche, die gut auf die Anforderungen der MAGIC-Analyse zugeschnitten ist. Allerdings hat es im Bereich der Teilchenseparation auch gewisse Schwächen. So können die verwendeten Separationsverfahren nicht ohne größeren Aufwand ausgetauscht und somit andere, eventuell besser geeignete Verfahren verwendet werden. Auch sind die Hillas-Parameter zwar gut für Teilchen mit Energien von über 60 Giga-Elektronenvolt GeV geeignet. Die Genauigkeit der Klassifikation nimmt aber bei niedrigeren Energien rapide ab.

Die vorliegende Arbeit nimmt auf den vorhandenen Parametern eine Merkmalsselektion vor und versucht weitere Parameter zu finden, die eine bessere Performanz zulassen. Dazu kombiniert ein evolutionärer Algorithmus die vorhandenen Parameter mit unterschiedlichen Rechenoperationen und wählt daraus mit Hilfe einer Bewertungsfunktion geeignete Merkmale aus.

Eine komfortable Plattform für die durchzuführenden Experimente bietet die Data-Mining-Software RapidMiner [19]. Sie bietet neben dem verwendeten evolutionärem Algorithmus auch viele unterschiedliche Klassifikations- und Validierungsverfahren. Damit umgeht sie die Beschränkung von MARS auf wenige vorgegebene Klassifikationsverfahren. Auch können die Ergebnisse leicht validiert werden.

Um einen Klassifikator zu trainieren, sind vorklassifizierte Daten notwendig. Dazu werden Monte-Carlo-generierte Gamma-Teilchen (s. Kap. 2.2.6) und Off-Daten (s. Kap. 2.1) verwendet.

Um diese Daten in RapidMiner analysieren zu können, müssen sie vom Binärformat von ROOT in ein für RapidMiner lesbares Format konvertiert werden. Teil 3 behandelt die eigens dazu erstellte Software. Natürlich müssen zunächst aus den vorliegenden Rohdaten mit Hilfe von MARS die Hillas-

1 **M**ajor **A**tmospheric **G**amma-Ray **I**maging **C**herenkov Telescopes

Parameter erzeugt werden. Um diesen Vorgang zu vereinfachen, wurden Programme entwickelt, die unter Zuhilfenahme von MARS ohne Benutzerinteraktion auch größere Datenmengen verarbeiten. Ein weiteres Programm wurde entwickelt, um die Bildparameter geeigneter Events aus dem ROOT-Format in das für RapidMiner lesbare CSV-Format zu konvertieren. Dabei ist die Programmstruktur darauf ausgelegt, dass mit geringem Aufwand auch die Extraktion anderer Parameter implementiert werden kann - unter Umständen sogar Zwischenergebnisse aus anderen Schritten in der MARS-Verarbeitungskette. Die bestehenden Auswahlkriterien für geeignete Events können unverändert wiederverwendet werden.

1.2 TERMINOLOGIE

In dieser Arbeit bezeichnet ein *Event* oder *Ereignis* das Eintreffen eines Teilchens, unabhängig davon, ob es es sich um ein Gamma-Teilchen oder ein hadronisches Teilchen handelt. Dies unterscheidet sich von der in der Physik gebräuchlichen Terminologie, die zwischen Excess Events (Gamma-Teilchen) und Hintergrund (Hadronen) unterscheidet. Jedes Event entspricht einem Beispiel im Sinne des Data Mining.

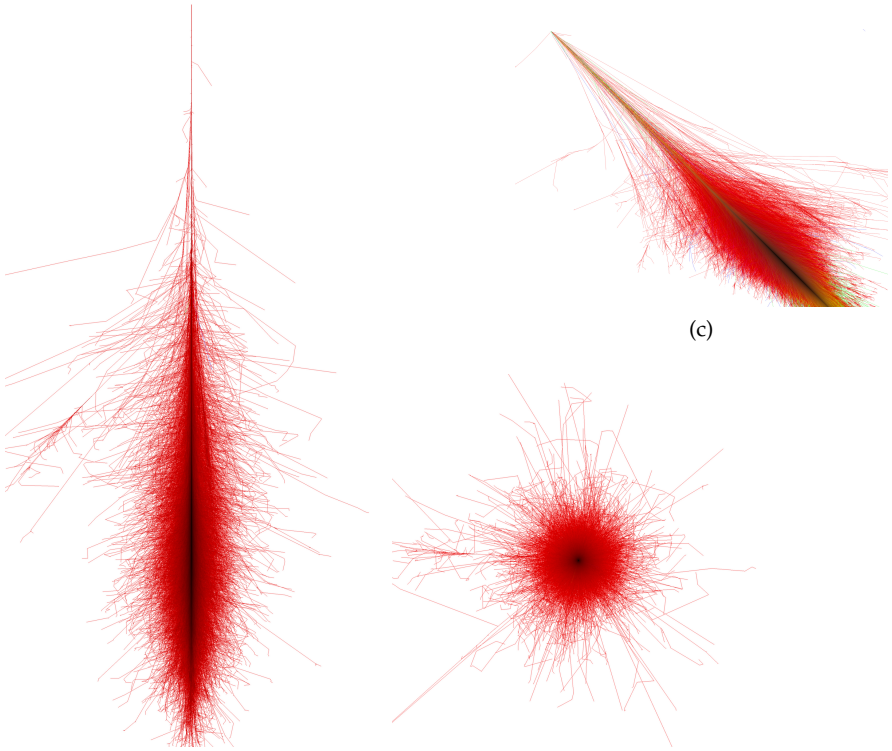
Im weiteren werden die vorhandenen Events *klassifiziert*, d.h. der Klasse „Gamma-Teilchen“ oder „Hadronen“ zugeordnet. Dies entspricht dem Prozess, der in der Physik in der Regel mit Separation, Hintergrundunterdrückung oder Signalextraktion bezeichnet wird.

Der Begriff *Parameter* wird in Anlehnung an bestehende Literatur synonym zu den aus dem Data Mining bekannten Begriffen *Merkmal* oder *Feature* gebraucht, da die derzeit in der MAGIC-Analyse gebräuchlichen Merkmale als Hillas-Parameter, Bildparameter etc. bezeichnet werden. Merkmale sind Größen, die ein Event charakterisieren, und anhand derer eine Klassifikation erfolgen kann.

1.3 PHYSIKALISCHE GRUNDLAGEN

Wenn entfernte Galaxien in ein zentrales schwarzes Loch kollabieren, werden große Energien frei, die in Form von Strömen hochenergetischer geladener Teilchen abgestrahlt wird - sogennanter *Jets*. Auch Pulsarwindnebel wie der Krebsnebel erzeugen solche Jets. Diese Jets enhalten insbesondere Gamma-Teilchen im Bereich von dutzenden GeV bis hin zu wenigen Terra-Elektronen-Volt TeV. Die Analyse dieser Teilchen erlaubt Rückschlüsse auf die Vorgänge im Kern solcher *aktiver Galaxien*.

Die Jets sind stark fokussiert, so dass sie auf der Erde nur dann beobachtet werden können, wenn der Jet genau in Richtung unseres Planeten zeigt. Treten die hochenergetischen Teilchen in die Erdatmosphäre ein, so wechselwirken sie mit Teilchen der Atmosphäre und lösen durch Kettenreaktionen ganze Lawinen aus, sogenannte *Teilchenschauer* (s. Abb. 1). Dabei entstehen kurze, schwache Lichtblitze, sogenanntes Cherenkov-Licht. Dieses wird vom MAGIC-Teleskop detektiert und aufgezeichnet. Dabei entstehen meist Ellipsenförmige Abbildungen, die Rückschlüsse auf die Eigenschaften des Teilchens erlauben.



(a) (b) (c)

Abbildung 1: Bilder von simulierten Luftschauern. (a) und (c) zeigen die Seitenansicht zweier Luftschauer; (b) ist die Projektion eines Luftschauers in die x-y-Ebene. (a) und (b) wurden von Teilchen erzeugt, das senkrecht in die Erdathmospäre eingedrungen sind. In (c) ist das Teilchen in einem Winkel von 45° in die Erdathmosphäre eingetreten. Quelle: [3]

Neben den Gamma-Teilchen treffen jedoch auch hadronische Teilchen, in erster Linie Protonen, auf die Erdatmosphäre und lösen Luftschauer aus. Der Anteil der Gamma-Teilchen an allen beobachteten Teilchen beträgt nur etwa 0,1 Prozent.

Diese Arbeit beschäftigt sich mit der Unterdrückung des hadronischen Hintergrundes anhand des aufgenommenen Cherenkov-Lichts.

1.4 ÜBERBLICK

Zunächst werden in Teil 2 die Grundlagen des bestehenden Analyseverfahrens in MAGIC erläutert sowie Ansatzpunkte für Verbesserungen aufgezeigt. Es werden Methoden des Data Mining beschrieben, mit deren Hilfe das bestehende Verfahren verbessert werden kann.

In Teil 3 werden Werkzeuge eingeführt, mit denen sich die MAGIC-Daten in ein Format konvertieren lassen, das für die Weiterverarbeitung mit den zuvor beschriebenen Verfahren geeignet ist.

Teil 4 nutzt die zuvor eingeführten Verfahren und Werkzeuge, um neue Merkmalssätze zu generieren und bessere Klassifikationsergebnisse zu erzielen. Die hier erhaltenen Ergebnisse werden in Teil 5 aufgeführt und bewertet.

Der vorliegende Bericht ist ein erster Schritt in Richtung auf eine flexiblere Arbeitsumgebung für systematische Untersuchungen der Eignung von Verfahren zu Merkmalsextraktion, -auswahl und -generierung sowie Data Mining für die astrophysikalische Analyse von Aufnahmen des MAGIC-Teleskops. Sie entstand als interdisziplinäre Studienarbeit von Marius Helf unter Betreuung von Katharina Morik (Informatik) und Wolfgang Rhode (Physik).

2 GRUNDLAGEN DER ANALYSE

In diesem Teil wird die Aufnahme der Daten, die Datenverarbeitung im Analyseprogramm MARS sowie die darin verwendeten Verfahren zur Parameterextraktion und Teilchen-Klassifikation beschrieben. Ein weiteres Kapitel setzt sich mit dem Programm CORSIKA auseinander, welches zur Simulation von Gamma-Events verwendet wird.

In Kapitel 2.3 werden außerdem weitere, im Rahmen dieser Arbeit verwendete Klassifikationsverfahren und Algorithmen erläutert.

2.1 AUFNAHME VON DATEN

2.1.1 *Aufbau des MAGIC-Teleskops*

Das MAGIC-Teleskop besteht aus einem beweglichen Reflektor mit 17 Metern Durchmesser, welcher das einfallende Licht auf eine Kamera mit einer Auflösung von 578 Pixeln in hexagonaler Anordnung bündelt. Jedes Pixel besteht aus einem Photomultiplier (PMT), der das einfallende Licht in eine Anzahl von Photoelektronen in der Einheit ph.e. umwandelt [5].

Soll eine Quelle beobachtet werden, wird das Teleskop auf die jeweilige Quelle ausgerichtet. Bei jedem einfallenden Lichtblitz wird das Aufnahmemodul des Teleskops getriggert, und für jeden Pixel der Kamera werden Intensität und Ankunftszeit des Lichts registriert. Bei der späteren Analyse ist zu beachten, dass sich die Bilder von im wesentlichen identischen Teilchen - abhängig von der Position am Himmel und weiteren Bedingungen - unterscheiden können. Auf diese Bedingungen wird in Kapitel 4.1.1 eingegangen.

2.1.2 *Trainingsdaten*

Für das Training eines Klassifikators zur Separation von Gammas und Hadronen müssen sowohl positive als auch negative Beispiele vorliegen, die unter möglichst identischen Bedingungen aufgenommen wurden. Es liegt in der Natur der Dinge, dass die Events aus Richtung Quelle noch nicht klassifizert sind - es kann sich sowohl um Hadronen als auch um Gammas handeln, da die Hadronen isotrop aus allen Richtungen in die Atmosphäre eindringen.

Der Simulator *CORSIKA* (s. Kap. 2.2.6) generiert Gamma-Events. Events, die lediglich Hadronen repräsentieren, erhält man durch zwei verschiedene Verfahren. Zum einen ist es möglich, die Kamera nicht nur auf die eigentliche Gamma-Quelle zu richten, sondern zeitweise auch auf einen Punkt in der Nähe der Quelle. Von dort können lediglich Hadronen in das Teleskop einfallen, so dass man die gewünschten Daten, sogenannte *Off-Daten* erhält, allerdings hat dieses Verfahren den Nachteil, dass in etwa der Hälfte der Zeit nicht die eigentlich interessante Quelle beobachtet werden kann (vgl. [8]).

Das zweite, zur Zeit meist eingesetzte Verfahren ist der sogenannte *Wobble-Mode* (s. Abb. 2, vgl. [8]). In diesem Aufnahmemodus wird die Kamera so ausge-

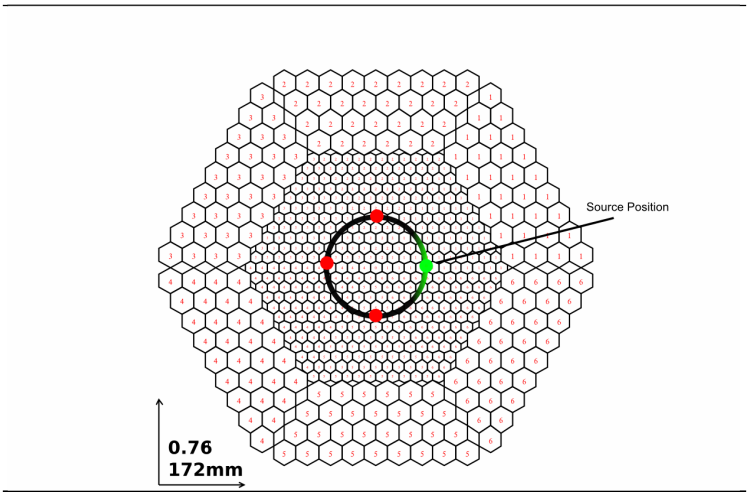


Abbildung 2: Wobble-Aufnahmemodus. Der grüne Punkt repräsentiert die tatsächliche Quellposition innerhalb der Kamera. Die drei roten Punkte geben die Position der Anti-Quellen an. Deutlich zu erkennen ist auch die hexagonale Anordnung der 578 Pixel.

richtet, dass sich die Quelle nicht genau im Zentrum, aber noch im Bildbereich der Kamera befindet. Um nun Off-Daten zu erhalten, bedient man sich der Tatsache, dass der Anteil der Gamma-Events nur 0.1 % - 1 % aller Events beträgt, und außerdem viele der Bildparameter abhängig von der Quellposition sind. Wählt man als Bezugspunkt für ihre Berechnung nicht die Position der Quelle, sondern den gegenüberliegenden Punkt der Kamera (*Anti-Source*), erhält man Parameter, die mit hinreichend großer Wahrscheinlichkeit ein Hadron beschreiben. Tatsächlich werden quellpositionsbezogene Parameter sogar für drei verschiedene virtuelle Quellpositionen berechnet. Im Falle unzureichender Performanz der Klassifikation ist die Tatsache, dass sich unter den negativen Beispielen bereits auch positive Beispiele befinden, jedoch ein Punkt, dem weitere Beachtung geschenkt werden sollte.

2.2 BESTEHENDES ANALYSEVERFAHREN

Zur Zeit wird zur Analyse der Teleskop-Daten die Analyse-Software MARS verwendet, sowie der Monte-Carlo-basierte Event-Generator CORSIKA. Beide Programmpakete werden im folgenden beschrieben.

2.2.1 *MARS*

MARS[1] besteht aus einer Verkettung von Programmen, die benutzt werden, um die Rohdaten, d.h. die von der Kamera aufgenommenen Intensitätswerte und Ankunftszeiten für jedes Pixel, schrittweise zu verarbeiten [8].

Zunächst kalibriert *Callisto*[2] die Rohdaten, um Hardwarefehler zu entdecken und Schwankungen in der Empfindlichkeit der Photo-Multiplier in der Kamera

1 MAGIC Analysis and Reconstruction Software
2 **Cal**ibrate **li**ght **s**ignals and **t**ime **o**ffsets

auszugleichen. In einem zweiten Schritt berechnet *Star*[3] aus den kalibrierten Daten die Bildparameter, die anschließend von *Ganymed*[4] zur Klassifikation der Events genutzt werden. Zuletzt rekonstruiert *Sponde*[5] die Energie der Gamma-Teilchen.

2.2.2 *Callisto: Kalibrierung*

Callisto kalibriert die Photomultiplier der Kamera, so dass individuelle Schwankungen der Verstärkungsfaktoren korrigiert sowie die Helligkeit des Himmels und dadurch verursachtes Rauschen, der sogenannte Night Sky Background (NSB), berücksichtigt werden. Außerdem kann Callisto defekte Pixel erkennen und durch Interpolation korrigieren. Dazu benötigt Callisto neben den eigentlichen Messdaten zusätzliche Daten, die vor der Messung aufgenommen werden müssen: die Calibration- und Pedestal-Daten. Die Calibration-Daten werden erzeugt, indem kurze Lichtpulse auf alle PMTs gesendet werden. Dadurch können der Umrechnungsfaktor von Licht in Photoelektronen sowie, durch den konstanten Abstand zwischen Lichtquelle und PMT, die Zeitverzögerung jedes PMTs berechnet werden.

Die Pedestaldaten dienen zur Kalibrierung des NSB. Bei ihnen handelt es sich um normale Aufnahmen, bei denen die Kamera auf eine quellfreie Position im selben Himmelssegment wie die Quelle gerichtet wird. Die Intensität des NSB ergibt sich sowohl aus natürlichen Einflüssen wie zum Beispiel dem Stand des Mondes, wie auch aus künstlichen Einflüssen wie Lichtverschmutzung durch Städte und andere Lichtquellen. [11]

Ausgabe von Callisto sind die kalibrierten und korrigierten Rohdaten in Form von Intensität und Ankunftszeit für jede einzelne PMT.

2.2.3 *Star: Merkmalsextraktion*

Die Hauptaufgabe von Star besteht in der Berechnung von über 30 Bildparametern aus den Rohdaten. Die Bilder von Schauern, die als Gamma-Events in Frage kommen, sind in etwa ellipsenförmig. Star passt eine Ellipse in die Bilder ein (s. Abb. 3) und berechnet aus Form und Lage der Ellipse die im nachfolgenden beschriebenen *Hillas-*, *Extended Hillas-*, *Image-* und *New Image-Parameter*. Abbildung 6 veranschaulicht einige dieser Parameter.

Zuvor wird eine Bildbereinigung vorgenommen, in der versucht wird, alle Pixel zu entfernen, die nicht zum eigentlichen Schauerbild gehören, sondern deren Lichtwerte durch Rauschen entstanden sind.

Datenbereinigung

Die Datenbereinigung befreit das Bild von Rauschen. Eine Möglichkeit hierzu ist das *Absolute Image Cleaning*, das allein anhand der Intensität eine Rauschunterdrückung vornimmt. Dazu wird jeder Pixel einer der drei Klassen Core-Pixel, benutzte und nicht-benutzte Pixel zugeordnet. Zunächst werden alle Pixel, die einen bestimmten, relativ hohen Intensitätswert erreichen oder überschreiten,

3 **St**andard **a**nalysis and **r**econstruction
4 **G**ammas **a**re **n**ow **y**our **m**ost **e**xciting **d**iscovery
5 **Sp**ectrum **on de**mand

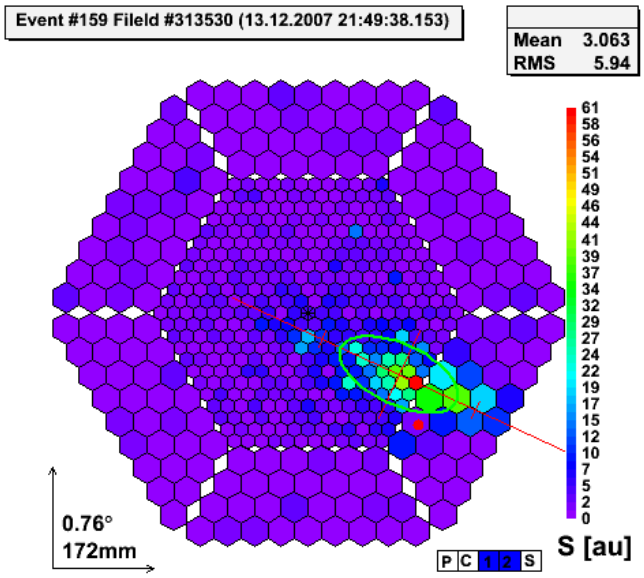


Abbildung 3: Signal der MAGIC-Kamera, Lichtintensität.

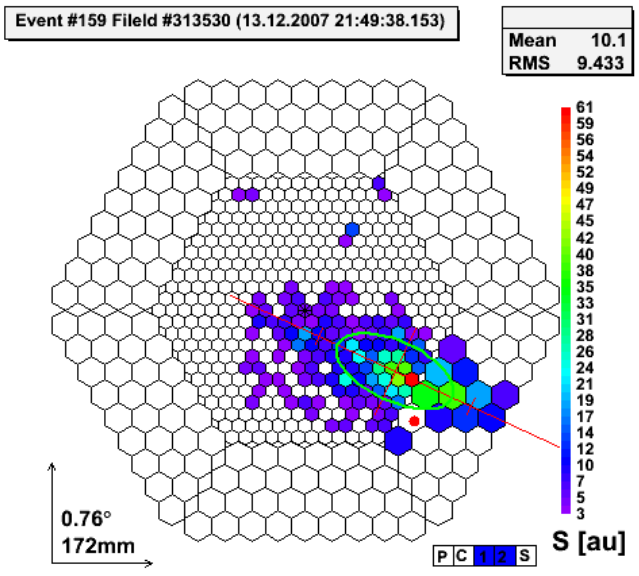


Abbildung 4: Bereinigtes Signal der MAGIC-Kamera, Lichtintensität. Deutlich zu erkennen sind nun das *MainIsland* und die *SubIslands*, sowie die von Star angepasste Ellipse.

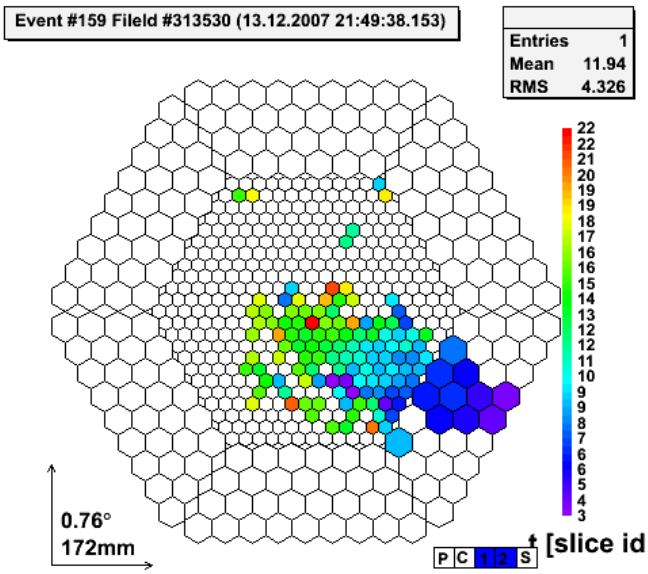


Abbildung 5: Bereinigtes Signal der MAGIC-Kamera, Zeitauflösung.

als Core-Pixel klassifiziert. Alle Pixel, die direkt mit einem Core-Pixel benachbart sind und eine weitere, etwas niedrigere Schwelle überschreiten, werden als benutzte Pixel definiert. Alle anderen Pixel sowie isolierte Core-Pixel werden entfernt, so dass nur zusammenhängende Pixel, sogenannte Inseln oder *Islands*, übrig bleiben, die eine gewisse Intensität aufweisen (s. Abb. 4).

Dieses Verfahren ist bei bei niederenergetischen Teilchen problematisch, bei denen viele Pixel nahe der Intensitätsgrenze liegen. Wird diese weiter herabgesetzt, ist die Wahrscheinlichkeit groß, dass auch Pixel, die nicht zum Signal gehören, nicht unterdrückt werden. Daher wird beim *Time Image Cleaning* die Ankunftszeit der Pixel betrachtet. Pixel, deren Ankunftszeit nicht in einem Zeitfenster von etwa 1.5 ns bis 2.0 ns liegt, werden unterdrückt. [4].

Klassische Hillas-Parameter

Die klassischen Parameter wurden 1985 von M.A. Hillas eingeführt (vgl. [15]):

SIZE Anzahl aller Photoelektronen im Bild. Diese ist proportional zum gesamten Licht im betrachteten Schauer.

WIDTH Kleine Halbachse der rekonstruierten Ellipse. Ein Maß für die horizontale Ausdehnung des Schauers.

LENGTH Große Halbachse der rekonstruierten Ellipse. Ein Maß für die vertikale Ausdehnung des Schauers.

COG *Center of Gravity* bzw. Mittelpunkt aller Pixel im Bild, gewichtet nach ihrer Intensität.

DIST *Distance*. Abstand des COG von der Quellposition in der Kamera.

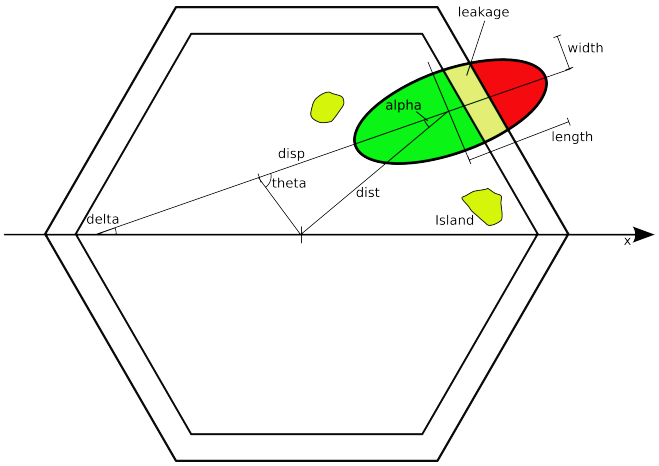


Abbildung 6: Verdeutlichung einiger Bildparameter (nach [11]).

ALPHA Winkel zwischen der großen Halbachse und der Verbindungslinie zwischen Quellposition und COG.

CONCN, CONCCOG, CONCCORE *Concentration*. Verhältnis der Photoelektronen der n hellsten Pixel zu SIZE, bzw. Verhältnis des COG oder der Core-Pixel zu SIZE.

LEAKAGEN Verhältnis der Photoelektronen in den n äußersten Pixel-Ringen zu SIZE. Maß für den Anteil des Schauers, der außerhalb des Bildbereichs der Kamera liegt. Ist LEAKAGE zu groß, fehlen Informationen zur Bestimmung der übrigen Bildparameter.

THETA, DISP Da Ellipsen aus Gamma-Events stets auf die Quellposition zulaufen, kann aus Orientierung und Größe der Ellipse eine fiktive Quellposition rekonstruiert werden, die in Verlängerung der großen Halbachse liegt. Disp gibt ihre Entfernung vom COG an und kann aus den Verhältnis der Halbachsen errechnet werden. Theta beschreibt den Winkel zwischen den Linien vom COG zur rekonstruierten Quellposition und von der beobachteten zur rekonstruierten Quellposition.

Weitere Bildparameter

Seit HILLAS die nach ihm benannten Hillas-Parameter vorgeschlagen hat, sind einige weitere Parameter entwickelt worden.

DELTA, SIN(DELTA), COS(DELTA) Delta entspricht dem Winkel zwischen der großen Halbachse und der x-Achse.

NUMISLANDS Anzahl der Inseln.

NUMSATPIXELSHG, NUMSATPIXELSLG Anzahl der Pixel, für die High Gain bzw. Low Gain gesättigt sind.

NUMUSEDPIXELS Anzahl der nach der Bildbereinigung verbleibenden Pixel.

NUMCOREPIXELS Anzahl der Core-Pixel.

NUMSINGLEPIXELS Anzahl der isolierten Einzel-Pixel, die nach dem ersten Schritt der Bildbereinigung noch vorhanden sind.

SIZESINGLEPIXELS Anzahl aller Photoelektronen der in NumSinglePixels gezählten PMTs.

USEDAREA, COREAREA Gesamtfläche aller benutzten bzw. aller Core-Pixel.

M3LONG, M3TRANS Dritte Momente entlang der großen bzw. kleinen Halbachse. Dienen dazu, dreiecksförmige Ereignisse auszusortieren.

SLOPELONG, SLOPETRANS Gradient im Zeitraum entlang der Halbachsen.

2.2.4 *Ganymed: Klassifikation*

Ganymed ordnet jedes Event einer der Klassen *Gamma* oder *Nicht-Gamma* zu. Zu letzterer gehören alle Hadronen sowie auch Events, über die nicht genügend Informationen vorliegen. Dazu nimmt Ganymed zunächst einige sogenannte Quality-Cuts vor, die Ereignisse aussortieren, welche nicht analysiert werden können. In einem zweiten Schritt wird ein Klassifikationsverfahren angewendet. In der Regel kommen Heuristiken in Form von weiteren Schritten zum Einsatz.

Nach der Klassifikation kommen weitere statistische Verfahren zum Einsatz, die unter Zuhilfenahme von Off-Daten falsch klassifizierte Events behandeln. Sie arbeiten jedoch nicht auf einzelnen Events, sondern nehmen Abschätzungen über die statistiche Verteilung der Gamma-Events vor.

Quality-Cuts

Die Quality-Cuts können in der Konfigurationsdatei von MARS angepasst werden. Standardmäßig werden folgende Cuts durchgeführt:

$$NumIslands < 3 \tag{1}$$

$$NumUsedPixels > 5 \tag{2}$$

$$Leakage1 < 0,3 \tag{3}$$

$$\log_{10}(ConcCOG) < -0,45 + 0,08 * (\log_{10}(Size) < 3,9) * (\log_{10}(Size) - 3,9)^2 \tag{4}$$

$$\log_{10}(Conc1) < -0,75 + 0,10 * (\log_{10}(Size) < 3,8) * (\log_{10}(Size) - 3,8)^2 \tag{5}$$

$$Length > -3,6 * (\log_{10}(Size) - 6,0)^2 + 70 \tag{6}$$

Events, die eine dieser Bedingung erfüllen, werden als Nicht-Gamma klassifiziert.

Es ist bekannt, dass Bilder von Gamma-Events nicht mehr als drei Inseln beinhalten (s. Gl. (1)). (2) und (3) entfernen Events, über die nicht genügend Informationen vorliegen, weil sie zu klein sind oder zu große Teile des Bildes außerhalb des Kamerabereichs liegen. Die übrigen drei Cuts entfernen weitere Events, die offensichtlich keine Gammas sind [8].

Cuts zur Separation

Die zur Separation der Teilchen entwickelten Heuristiken, also weitere Cuts, können unverändert für alle Datensätze eingesetzt werden können. Sie machen sich zu Nutze, dass die große Halbachse von Gamma-Events immer in Richtung der Quelle zeigt. Somit können Events mit großem Theta ausgeschlossen werden. Ein weiterer Cut setzt Area und Size in Relation (vgl. [8]):

$$\theta^2 < c_0^2 \tag{7}$$

$$A < c_3 \cdot (1 - c_4 \cdot (\log_{10}(S) - c_5)^2 \tag{8}$$

Dabei steht A für Area und S für Size. Die c_i sind freie Parameter, die in der Konfigurationsdatei von Ganymed angegeben werden.

Die Performanz der Cuts ist in der Regel trotz ihrer Einfachheit relativ gut, kann jedoch, wie in Teil 5 gezeigt wird, durch Verwendung geeigneter Merkmalssätze und Klassifikationsverfahren weiter gesteigert werden.

2.2.5 *Sponde: Energierekonstruktion*

Nachdem die Ereignisse von Ganymed klassifiziert und quantifiziert wurden, schätzt Sponde mit Hilfe ausgewählter Merkmale und einer Random Forest Regression ([7]) die Energie der Ereignisse.

2.2.6 *CORSIKA: Monte-Carlo-Generierung von Gamma-Ereignissen*

Um die von Ganymed und Sponde benutzten Random Forests sowie die weiter unten beschriebenen Klassifikatoren zu trainieren, sind vorklassifizierte Trainingsmengen erforderlich. Muster für Hadronen sind in Form von Off-Daten leicht zu erhalten. Es ist hingegen nicht möglich, eine Menge von reinen Gamma-Events zu beobachten, da der Anteil des hadronischen Hintergrundes den Anteil der Gamma-Events um drei Größenordnungen übersteigt. Daher müssen Referenzdaten simuliert werden.

CORSICA[6] [14] ist in der Lage, solche Simulationen durchzuführen. Es ist nicht Teil des MARS-Paketes, sondern ein eigenständiges, am Forschungszentrum Karlsruhe entwickeltes Programm. Teilchenschauer, die von kosmischen Teilchen erzeugt werden, können detailliert simuliert werden. Dazu stehen verschiedene Modelle mit unterschiedlichen Anwendungsgebieten, Genauigkeit und Rechenaufwand zur Verfügung, unter denen der Benutzer wählen kann. Da für die Luftschauer keine analytische Beschreibung existiert, sondern es sich vielmehr um statistische Prozesse handelt, für die lediglich die Verteilungen bekannt sind oder abgeschätzt werden können, bedient sich CORSIKA der Monte-Carlo-Methode. Die Wechselwirkungen der kosmischen Teilchen mit Teilchen

6 Cosmic Ray Simulation for Kascade

in der Erdathmosphäre sowie die Lebensdauer der erzeugten Teilchen wird unter Berücksichtigung der gegebenen Verteilungen ausgewürfelt [13].

CORSIKA beschreibt lediglich die Luftschauer an sich, nicht die von ihnen erzeugte Cherenkov-Strahlung. Um die Daten von CORSIKA in MARS verwenden zu können, sind deshalb weitere Verarbeitungsschritte notwendig, die zuletzt zu einer simulierten Aufnahme des erzeugten Cherenkov-Lichts durch die MAGIC-Kamera in dem von MARS verwendeten Datenformat führen.

2.3 VERBESSERUNG DES ANALYSEVERFAHRENS

Die Verbesserung des Analyseverfahrens setzt bei der Klassifikation anhand der Bildparameter an. Anstelle des Random Forests bzw. der Cuts in Ganymed soll die Eignung weiterer Klassifikatoren untersucht werden, wie zum Beispiel *Support Vector Machines* (SVM), *Nearest Neighbours* und weitere. Diese haben vor allem gegenüber den Cuts den Vorteil, für verschiedenartige Eingabedaten trainiert und angepasst werden zu können, so dass sie für verschiedene Beobachtungsbedingungen optimiert werden können. Unter Umständen ergibt sich bereits durch die Wahl neuer Klassifikatoren eine Verbesserung der Performanz.

Zusätzlich sollen auf Basis der vorhandenen Parameter neue Merkmale generiert werden. Dazu wird ein evolutionärer Algorithmus verwendet.

2.3.1 *RapidMiner*

Für Aufbau und Durchführung der Experimente ist eine Umgebung vorteilhaft, die es mit geringem Aufwand ermöglicht, verschiedene Klassifikatoren, Merkmalsgeneratoren und andere Bausteine zusammenzusetzen und auszutauschen. Dies leistet die Data Mining-Software RapidMiner [19]. RapidMiner ist eine Java-basierte IDE[7] für Data Mining. Die Entwicklung wurde 2001 am Lehrstuhl für künstliche Intelligenz an der Universität Dortmund unter dem Namen YALE begonnen. In einer GUI-Oberfläche können Prozesse aus Einzelbausteinen entwickelt und zusammengesetzt werden, ohne dass die verwendeten Verfahren neu implementiert werden müssen. Die Prozesse werden im XML-Format abgespeichert und können wahlweise von der GUI oder von einer Konsolenapplikation evaluiert werden.

Um die Daten des MAGIC-Teleskops in RapidMiner verarbeiten zu können, müssen sie vom binären ROOT-Format in ein von RapidMiner unterstütztes Format konvertiert werden. Die hierzu notwendigen Schritte werden in Teil 3 beschrieben.

In den nun folgenden Kapiteln werden einige der in RapidMiner zur Verfügung stehenden Lernverfahren und andere Algorithmen vorgestellt, die im Rahmen dieser Arbeit (vgl. Teil 4) zum Einsatz kommen.

2.3.2 *SVM*

Die Support Vector Machine, kurz SVM, ist ein Lernverfahren, das auf VAPNIK ([22]) zurückgeht. Sie wird für binäre Klassifikationsaufgaben oder zur Re-

7 Integrated Development Environment

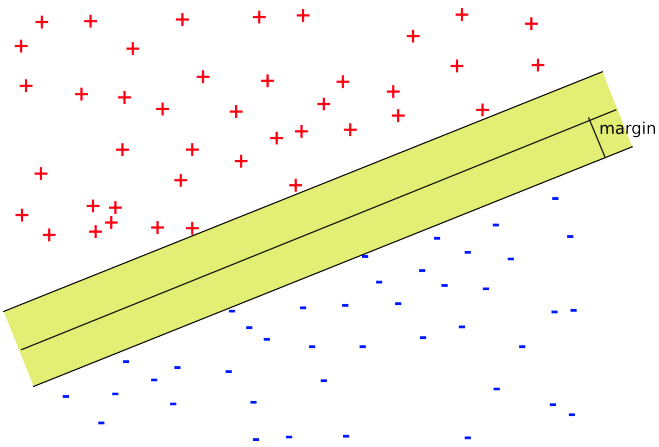


Abbildung 7: Optimale trennende Hyperebene im separierbaren Fall. Keine Beispiele liegen auf dem Margin.

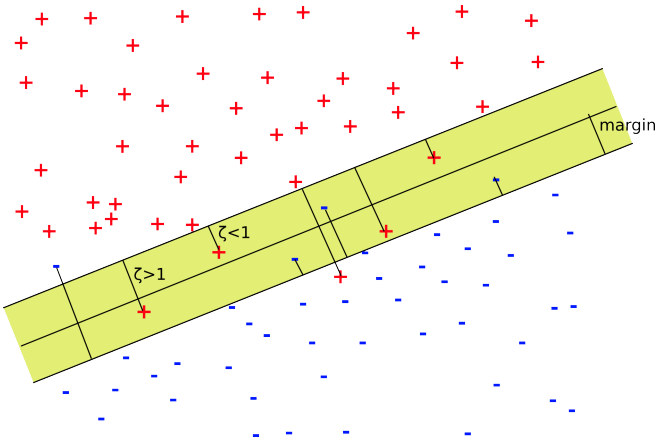


Abbildung 8: Optimale trennende Hyperebene im nicht-separierbaren Fall. Einige Beispiele liegen auf dem Margin und auf der falschen Seite der trennenden Hyperebene. Diese werden durch Slack-Variablen bestraft.

gression genutzt. Daneben gibt es auch Varianten für strukturelle Modelle (bei denen für eine Beobachtung $\vec{x}$ nicht eine Zahl, sondern ein Vektor $\vec{y}$ ausgegeben wird) und für Clustering. Für die Separation von Gammas und Hadronen wollen wir zunächst die Klassifikation von der SVM lernen lassen.

Sei die Trainingsmenge gegeben als N Tupel $(\vec{x}_i, y_i)$, wobei $y_i \in \{-1; 1\}$ das Label des Beispiels x_i angibt. Geht man zunächst davon aus, dass die Daten linear separierbar sind, so lässt sich eine Hyperebene $\{\vec{x} : f(x) = \vec{x} \cdot \vec{\beta} + \beta_0 = 0\}$ bestimmen, die den Merkmalsraum so in zwei Halbräume unterteilt, dass auf der einen Seite nur positive, auf der anderen nur negative Beispiele liegen.

Wählt man $\vec{\beta}'$ so, dass $|\vec{\beta}'| = 1$, dann kann eine Klassifikation erfolgen durch (vgl. [12]):

$$G(x) = \text{sign}(\vec{x} \cdot \vec{\beta}' + \beta'_0) \tag{9}$$

Dabei gibt das Argument des sign-Operators den vorzeichenbehafteten Abstand des Beispiels $\vec{x}$ von der Hyperebene an. Die optimale trennende Hyper-

ebene liegt so, dass sowohl der Abstand M zum nächsten positiven Beispiel als auch zum nächsten negativen Beispiel maximal ist. Abb. 7 skizziert eine solche Ebene. Der Raum ohne Beispiele zwischen der Hyperebene und den nächsten Beispielen heißt *Margin*. Wenn kein Beispiel auf dem Margin liegt, so muss gelten (vgl. [12])

$$\vec{y}_i(\vec{x}_i \cdot \vec{\beta}' + \beta_0') \geq M,\ i = 1, \ldots, N\,, \tag{10}$$

und M soll maximiert werden:

$$\max_{\vec{\beta}', \beta_0', |\vec{\beta}'|=1} M\,. \tag{11}$$

Teilt man diese Gleichung durch M, und wählt $\vec{\beta} = \frac{\vec{\beta}'}{M}$ und $\beta_0 = \frac{\beta_0'}{M}$, so erhält man

$$\vec{y}_i(\vec{x}_i \cdot \vec{\beta} + \beta_0) \geq 1,\ i = 1, \ldots, N\,, \tag{12}$$

und es soll $|\vec{\beta}|$ minimiert werden:

$$\min_{\vec{\beta}, \beta_0} |\vec{\beta}|\,. \tag{13}$$

Im allgemeinen Fall sind die zu verarbeitenden Daten aber nicht vollständig separierbar. In diesem Fall müssen Ausnahmen in Form von Beispielen, die innerhalb des Margins und unter Umständen sogar auf der falschen Seite der Hyperebene liegen, zugelassen werden (vgl. Abb. 8). Formell wird dies durch die Einführung von *Slack-Variablen* (vgl. [12]) $\xi_i \in \mathbb{R}^+$ erreicht. ξ_i gibt an, wie weit ein Beispiel innerhalb des Margins liegt, in Einheiten der Breite des Margins $\frac{1}{M}$. Ein Beispiel x_i wird somit genau dann fehlklassifiziert, wenn $\xi_i > 1$. Die Summe aller ξ_i ist durch einen frei wählbaren Parameter c nach oben beschränkt. Somit können nicht mehr als c Beispiele fehlklassifiziert werden. Gleichungen (12) und (13) ergeben sich zu

$$\begin{aligned} &\min_{\vec{\beta}, \beta_0} |\vec{\beta}| \\ \text{so dass}\quad &\forall i : \vec{y}_i(\vec{x}_i\vec{\beta} + \beta_0) \geq 1 - \xi_i,\ \xi_i \geq 0,\ \sum \xi_i < c\,. \end{aligned} \tag{14}$$

Dies ist äquivalent zu

$$\begin{aligned} &\min_{\vec{\beta}, \beta_0} \frac{1}{2}\vec{\beta}^2 \\ \text{so dass}\quad &\forall i : \vec{y}_i(\vec{x}_i\vec{\beta} + \beta_0) \geq 1 - \xi_i,\ \xi_i \geq 0\,. \end{aligned} \tag{15}$$

Hier wurde die Konstante c durch einen Kostenfaktor C ersetzt.

Nach dem Prinzip der quadratischen Programmierung ergibt sich daraus das primale Problem in Lagrange-Formulierung ([12]):

$$L_P = \frac{1}{2}\vec{\beta}^2 + C\sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \alpha_i[y_i(\vec{x}_i\vec{\beta} + \beta_0) - (1 - \xi_i)] - \sum_{i=1}^{N} \mu_i\xi_i\,. \tag{16}$$

Dabei sind die μ_i und α_i die Lagrange-Faktoren. L_P soll nun nach $\vec{\beta}$, β_0 und den ξ_i minimiert werden. Dazu werden die Ableitungen zu Null gesetzt, woraus sich die folgenden Gleichungen ergeben:

$$\vec{\beta} = \sum_{i=1}^{N} \alpha_i y_i x_i \tag{17}$$

$$0 = \sum_{i=1}^{N} \alpha_i y_i \tag{18}$$

$$\forall i : \alpha_i = C - \mu_i \tag{19}$$

Mit Hilfe dieser Gleichungen erhält man aus dem primalen Problem das duale Problem

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \cdot f(\vec{x}_i, \vec{x}_j) \tag{20}$$

mit $f(\vec{x}_i, \vec{x}_j) = \vec{x}_i \cdot \vec{x}_j$. Der Vorteil dieser Umformung liegt auf der Hand, wenn man sich vor Augen führt, dass das duale Problem nur N freie Parameter in Form der α_i hat, während das primale Problem mit den μ_i und xi_i weitere $2N$ Parameter besitzt. Dies verringert den Aufwand für die Optimierung erheblich.

L_D wird unter Einhaltung von Gl. (17), (18) und (19) sowie Gl. (21), (22) und (23), die sich durch weitere Umformungen ergeben, maximiert. Dies ist analytisch nicht möglich, es existieren jedoch effiziente numerische Verfahren, wie SMO (Sequential Minimal Optimization, vgl. [12]) oder evolutionäre Algorithmen [18].

$$\alpha_i [y_i (f(\vec{x}_i, \vec{\beta}) + \beta_0) - (1 - \xi_i)] = 0 \tag{21}$$

$$\mu_i \xi_i = 0 \tag{22}$$

$$y_i (f(\vec{x}_i, \vec{\beta}) + \beta_0) - (1 - \xi_i) \geq 0 \tag{23}$$

Nachdem die α_i mit einer geeigneten Methode bestimmt wurden, kann das optimale $\hat{\vec{\beta}}$ nach Gl. (17) berechnet werden; es ist also eine Linearkombination aus Beispiel-Vektoren. Aus Gl. (21) und (23) geht hervor, dass nur solche Vektoren einen endlichen Beitrag dazu leisten, die auf dem Margins liegen (vgl. [12]). Diese Vektoren heißen *Support-Vektoren* und sind die Namensgeber der SVM.

Kernfunktionen

In der oben vorgestellten Form liefert die SVM nur für linear weitgehend separierbare Datensätze gute Ergebnisse. Ist der optimale Schnitt zwischen den Klassen nicht linear, kann offensichtlich keine gute trennende Hyperebene berechnet werden. Unter Umständen kann aber eine Transformation des Merkmalsraums mittels einer vektoriellen Funktion $\vec{h}(\vec{x})$ zu besseren Ergebnissen führen. Im vorherigen Abschnitt wurde die Funktion $f(\vec{x}_i, \vec{x}_j)$ eingeführt, die in der klassischen Version der SVM das Skalarprodukt zweier Merkmalsvektoren berechnet. Setzt man nun anstelle der Merkmalsvektoren die transformierten Merkmalsvektoren ein, ergibt sich

$$f(\vec{x}_i, \vec{x}_j) = \vec{h}(\vec{x}_i) \cdot \vec{h}(\vec{x}_j)\,. \tag{24}$$

Das Skalarprodukt wird also in einem anderen Raum berechnet. Es ist aber auch möglich, anstelle des Skalarprodukts andere sogenannte *Kernfunktionen* zu wählen, die die explizite Transformation der einzelnen Beispiele umgehen. Die einzige Bedingung an die Kernfunktionen ist, dass sie die Merkmale in einen Hilbert-Raum transformieren, in dem per definitionem ein Skalarprodukt existiert. Durch Wahl beispielsweise radialer oder polynomieller Kernfunktionen können dann unter Umständen auch nicht linear separierbare Daten effektiv klassifiziert werden.

2.3.3 *k-Nearest-Neighbors*

k-Nearest-Neighbors, kurz kNN, ist ein relativ einfaches Klassifikationsverfahren, das auch Mehrklassenprobleme lösen kann (vgl. [12]). Die Grundgedanken wurden bereits 1951 von FIX und HODGES beschrieben [10]. Im Gegensatz zur SVM, das die gesamte Trainingsmenge nutzt um ein Modell zu berechnen, nutzt kNN zur Klassifikation unbekannter Beispiele nur wenige bekannte Beispiele aus der direkten Umgebung des zu klassifizierenden Beispiels und wird deshalb zu den lokalen Lernverfahren gezählt.

Die Trainingsmenge sei wiederum gegeben als Menge vorklassifizierter Tupel $(\vec{x}_i, y_i)$. Im Merkmalsraum muss weiterhin ein Distanzmaß $d(\vec{x}_i, \vec{x}_j)$ definiert sein. Für numerische Merkmale kommen beispielsweise das euklidsche Distanzmaß oder die Manhattan-Distanz in Frage. Soll nun ein unbekanntes Beispiel $\vec{x}_0$ klassifiziert werden, bestimmt kNN bezüglich des gewählten Distanzmaßes die k nächsten Nachbarn von $\vec{x}_0$. Die vorhergesagte Klasse wird per Mehrheitsentscheid bestimmt. Unter Umständen können die Nachbarn nach einem Ähnlichkeitsmaß zu x_0 gewichtet werden. Bei Gleichstand wird das Ergebnis ausgwürfelt. Für ein Zweiklassenproblem ist der Schätzer für die Klasse eines unbekannten Beispiels $\vec{x}$

$$\hat{f}(\vec{x}) = \frac{1}{k} \sum_{x_i \in N_k(\vec{x})} w_i y_i \,. \tag{25}$$

Dabei ist N_k die Menge der k nächsten Nachbarn und w_i das Gewicht des i-ten Nachbarn, beispielsweise $w_i = \mathrm{Sim}(\vec{x}, \vec{x}_i)$, wobei $\mathrm{Sim}(\vec{x}, \vec{x}_i)$ über das verwendete Distanzmaß definiert werden kann.

Neben dem Distanzmaß und der Berechnung der Gewichte ist die Anzahl k der betrachteten Nachbarn der einzige Parameter des Verfahrens. Für $k = 1$ neigt kNN zu Overfitting, da jedes einzelne Trainingsbeispiel die Klassengrenze stark beeinflusst. Wird k zu niedrig gewählt, generalisiert das Verfahren in der Regel schlecht. Wird k andererseits zu groß gewählt, entsteht ein Underfitting und die Beispiel-Klassen werden unter Umständen nicht hinreichend genau beschrieben. Eine Möglichkeit, einen geeigneten Wert für k zu finden, stellt die Kreuzvalidierung dar (s. Kap. 2.3.4).

kNN ist trotz seiner simplen Funktionsweise häufig erfolgreich, insbesondere dann, wenn Beispiele einer Klasse sehr unterschiedliche Ausprägungen haben, oder die Grenze der Klasse sehr unregelmäßig verläuft. In diesem Fall hätte beispielsweise eine SVM Schwierigkeiten, eine geeignete Hyperebene zu finden.

2.3.4 *Kreuzvalidierung*

Die Kreuzvalidierung ist ein einfaches, aber effektives Mittel, die Eignung eines gelernten Modells (hier ist der Begriff nicht nur auf Merkmale beschränkt) für eine Klassifikationsaufgabe abzuschätzen. Eine Trainingsmenge mit N Beispielen wird zufällig in n disjunkte Teilmengen aufgeteilt. Unter Auslassung der i-ten Teilmenge wird ein Modell auf den Trainingsdaten erzeugt und auf der ausgelassenen Menge angewendet. Dies wird für alle n Teilmengen wiederholt und jeweils die Klassifikationsgenauigkeit p_i bestimmt. Durch Mittelung über alle p_i erhält man eine Abschätzung für die Güte des Modells und damit auch der verwendeten Merkmale.

Die Kreuzvalidierung hilft bei der Modellselektion hinsichtlich der Programmparameter einer einzelnen Modellklasse (z.B. die Anzahl der nächsten Nachbarn bei kNN), oder auch die zur Klassifikation verwendeten Merkmale. Somit spielt die Kreuzvalidierung eine zentrale Rolle bei vielen Problemstellungen des Data Mining.

2.3.5 *Gütemaße*

Die Qualität einer Klassifikation wird anhand eines Gütemaßes gemessen. In dieser Arbeit werden die Merkmalssätze bezüglich der *Accuracy* optimiert. Zur ihrer Berechnung wird die Anzahl der *True Positives* (TP) und *True Negatives* (TN) verwendet. TPs sind solche Beispiele, die zur Klasse der positiven Beispiele (Gammas) gehören, und auch als solche klassifiziert wurden. TNs sind analog dazu Beispiele, die zur Klasse der negativen Beispiele gehören und als negativ klassifiziert wurden. Weiterhin gibt es *False Positives* und *False Negatives*, die in Wirklichkeit zur Klasse der negativen (bzw. positiven) Beispiele gehören, aber als positiv (bzw. negativ) klassifiziert wurden (s. Tab. 1).

	Vorhersage		
	True	False	
True	TP	FN	Recall
False	FP	TN	
	Precision		

Tabelle 1: Veranschaulichung von Precision und Recall

Die *Accuracy* gibt die Wahrscheinlichkeit an, dass die Klassifikation eines Beispiels die tatsächliche Klasse liefert und wird abgeschätzt als

$$Acc = \frac{\#TP + \#TN}{\#total}. \tag{26}$$

Weiterhin werden *Precision* und *Recall* betrachtet:

$$Prec = \frac{\#TP}{\#TP + \#FP} \tag{27}$$

$$Rec = \frac{\#TP}{\#TP + \#FN} \tag{28}$$

Precision und Recall beziehen sich auf jeweils auf die Klasse der positiven Beispiele. Precision gibt die Wahrscheinlichkeit an, dass ein als positiv klassifiziertes Beispiel tatsächlich in der positiven Klasse liegt, während Recall die Wahrscheinlichkeit angibt, dass ein positives Beispiel auch als solches erkannt wird.

Precision und Recall können analog für die negative Klasse berechnet werden.

Die positiven bzw. negativen Beispiele können gewichtet werden, wenn beispielsweise in der Realität der Anteil der negativen Beispiele höher ist als in der Testmenge. Sollen negative Beispiele f-mal stärker gewichtet werden, dann ergibt sich die Accuracy zu

$$Acc_w = \frac{\#TP + f \cdot \#TN}{(\#TP + \#FP) + f \cdot (\#TN + \#FN)} . \tag{29}$$

Gleichung (27) wird zu

$$Prec_w = \frac{\#TP}{\#TP + f \cdot \#FP} . \tag{30}$$

2.3.6 *Merkmalsgenerierung und -selektion*

Oft beinhaltet ein Merkmalssatz viele Merkmale, die redundante Informationen enthalten oder irrelevant für die Klassifikationsaufgabe sind. Zudem können Beziehungen zwischen ihnen bestehen, die von Klassifikatoren nicht erkannt werden können. Für ein gutes Klassifikationsergebnis ist es jedoch zwingend erforderlich, dass geeignete Merkmale existieren. Redundanz und irrelevante Merkmale können das Ergebnis stark verschlechtern, während gut formulierte Beziehungen es erheblich verbessern können. Hinzu kommt der *Fluch der hohen Dimension* [12]: fast jedes Lernverfahren hat Einbußen in Laufzeit oder Klassifikationsergebnis, wenn der Merkmalsraum zu hochdimensional ist. Abhilfe versucht die *Merkmalsselektion* zu schaffen, die sich mit der Auswahl einer geeigneten Teilmenge der Merkmale beschäftigt, während die *Merkmalsgenerierung* nach verborgenen Beziehungen sucht [17].

Bei der Merkmalsselektion werden systematisch Teilmengen der Merkmalsmenge ausgewählt und beispielsweise mit Hilfe der Kreuzvalidierung bewertet.

Die Merkmalsgenerierung sucht nach Beziehungen zwischen den Merkmalen, beispielsweise könnte aus den Merkmalen *Width* und *Length* durch Multiplikation das Merkmal *Area* konstruiert werden. Mit Hilfe der Merkmalsselektion muss nun wiederum aus neuen und alten Merkmalen ein geeigneter Merkmalssatz zusammengestellt werden. Konstruktion und Selektion gehen also miteinander einher.

Implementierungen zur Merkmalsgenerierung und -Selektion bedienen sich unter anderem evolutionärer Algorithmen, wie sie auch in dieser Arbeit verwendet und im folgenden Kapitel beschrieben werden.

```
1  BEGIN
2    INITIALISE population with random candidate solutions;
3    EVALUATE each candidate;
4    REPEAT UNTIL ( TERMINATION CONDITION is satisfied ) DO
5      SELECT parents;
6      CROSSOVER pairs of parents;
7      MUTATE the resulting offspring;
8      EVALUATE new candidates;
9      SELECT individuals for the next generation;
10   OD
11 END
```

Listing 2.1: Der Ablauf eines evolutionären Algorithmus in Pseudocode (nach [9])

2.4 EVOLUTIONÄRE ALGORITHMEN

Evolutionäre Algorithmen (vgl. [9]), kurz EA, ahmen in gewisser Weise den DARWIN'schen Evolutionsprozess nach. Eine Population von Individuen wird schrittweise durch Variation und Auslese an eine Umwelt angepasst. Dabei spielen *Selektion*, *Mutation* und *Kreuzung* von Individuen eine zentrale Rolle.

Ein EA verfügt anschaulich über einen initialen Genpool, eine *Population* aus mehreren Individuen, die daraufhin überprüft werden, inwieweit sie an die Umwelt angepasst sind. Der Algorithmus erzeugt iterativ Generationen von Individuen. Beim Generationswechsel werden die angepasstesten Individuen mit hoher Wahrscheinlichkeit selektiert und dann mutiert und gekreuzt, um anschließend erneut dem Genpool zugeführt zu werden, während schlecht angepasste Individuen den Generationswechsel wahrscheinlich nicht überleben. Listing 2.1 veranschaulicht diesen Vorgang.

In der Informatik bestehen die Individuen klassischerweise aus Bitstrings, die ein Genom repräsentieren. Jedes Bit stellt ein einzelnes Gen dar, dass entweder aktiviert (1) oder deaktiviert (0) ist. Die Umwelt ist durch eine bestimmte Problemstellung gegeben. Eine Fitness-Funktion $f(x)$ liefert ein Maß dafür, wie gut ein Individuum x an die jeweilige Umwelt angepasst ist, bzw. wie gut es sich zur Lösung des Problems eignet. Die Population hat eine feste Größe N. Durch Variationen ihrer Individuen sollen neue, bessere Individuen generiert werden.

2.4.1 *Ablauf eines evolutionären Algorithmus'*

Ein EA besteht aus einer Initialisierungsphase, sowie einer Schleife, in der Elternselektion, Mutation, Crossover und Selektion überlebender Individuen ausgeführt werden. Sobald eine Abbruchbedingung erfüllt ist, terminiert der Algorithmus. Jeder Durchlauf der Schleife entspricht einer Generation. Häufige Abbruchbedingungen sind das Durchlaufen einer bestimmte Anzahl Generation oder Fitness-Auswertungen, das Erreichen einer bestimmte Fitness oder das Erzeugen einer bestimmten Anzahl von Generationen, in den keine Verbesserung festgestellt werden konnte.

INITIALISIERUNG Während der Initialisierungsphase wird zunächst eine Population mit zufällig erzeugten Individuen gefüllt: der EA generiert N Bitstrings, deren Bits zufällig auf 0 oder 1 gesetzt werden.

ELTERNSELEKTION Die Elternselektion ist der Prozess der Auswahl solcher Individuen, die mutiert oder gekreuzt werden. Meist werden starke, d.h. gut angepasste Individuen mit einer hohen Fitness f gewählt. Die Elternselektion wählt also Individuen, die Eltern der nächsten Generation werden sollen, auf Grundlage ihrer Qualität aus. Allerdings erhalten auch schwächere Individuen eine kleine Wahrscheinlichkeit, als Eltern ausgwählt zu werden. Anderenfalls wäre die Wahrscheinlichkeit groß, dass lediglich ein lolakes Optimum gefunden wird.

CROSSOVER Der Crossoveroperator ist ein binärer Operator und kombiniert zwei Individuen mit der Intention neue, besser angepasste Individuen zu generieren. Es gibt mehrere sogenannte Rekombinationsverfahren. Ein weit verbreitetes ist das *Single-Point-Crossover*. Dabei wird zufällig eine Position innerhalb der Bitstrings bestimmt und alle Bits rechts von dieser Position zwischen den beiden Individuen ausgetauscht.

MUTATION Der Mutationsoperator ist ein unärer Operator. Er erhält genau ein Individuum als Eingabe und variiert es gemäß einer stochastischen Vorschrift. Bei Bitstrings werden in der Regel mit einer Wahrscheinlichkeit p_m einzelne Bits geflippt. Der Mutation kommt ein wichtiger Punkt zu, der bei alleiniger Verwendung des Crossoveroperators nicht gegeben wäre: er gewährleistet, dass jeder Punkt des Lösungsraumes erreicht werden kann, da jedes Bit mit einer Wahrscheinlichkeit größer Null jeden Wert annehmen kann. Bei alleiniger Verwendeung von Crossover-Variationen könnte ein Bit, dass in allen Individuen deaktiviert ist, nicht aktiviert werden. Somit trägt der Mutationsoperator auch dazu bei, dass die Population sich nicht um ein einziges lokales Maximum herum verdichtet, welches nicht notwendigerweise das globale Maximum ist.

SELEKTION ÜBERLEBENDER INDIVIDUEN In dieser Phase werden wiederum auf Grundlage der Fitness Individuen ausgewählt, die in die Population der nächsten Generation übernommen werden. Sie entspricht im wesentlichen der Elternselektion, wird aber in einer anderen Phase des Algorithmus durchgeführt.

Bei dieser Selektion werden alle Individuen, sowohl die alten als auch der durch Variationen entstandenen neuen Individuen, berücksichtigt.

2.4.2 *Evolutionäre Algorithmen zur Merkmalsgenerierung und -Selektion*

Das vorherige Kapitel beschreibt die prinzipielle Vorgehensweise von evolutionären Algorithmen am klassischen Beispiel von Bitstrings. Evolutionäre Algorithmen zur Merkmalsgenerierung und -selektion [20] funktionieren nach dem gleichen Prinzip, jedoch muss eine geeignete Repräsentation des Problems gefunden werden: die Population besteht aus N Individuen, die jeweils einen Merkmalssatz beschreiben. Ein Individuum x beinhaltet eine Liste von Merk-

```
create an initial population
evaluate initial population
repeat
  perform selection
  perform variable-length crossover
  perform mutation
  perform feature generation
  evaluate population
until termination criterion is fulfilled
```

Listing 2.2: Der Ablauf eines evolutionären Algorithmus zur Merkmalsgenerierung und -Selektion in Pseudocode (nach [20]). Der Unterschied zum klassischen evolutionären Algorithmus besteht im wesentlichen in der neu hinzugekommenen Merkmalsgenerierung.

malen und zu jedem Merkmal einen Vermerk, der angibt, ob das Merkmal berücksichtigt werden soll.

Die Mutations- und Crossoveroperatoren müssen für diese Repräsentation angepasst werden. Hinzu kommt ein Merkmalsgenerator, der vor der Selektion überlebender Individuen angesetzt wird (vgl. Listing 2.2). Außerdem wird eine geeignete Fitness-Funktion beschrieben.

FITNESS-FUNKTION Die Fitnessfunktion $f(x)$ bewertet ein Individuum x anhand der Klassifikationsgenauigkeit eines beliebigen, zuvor festgelegten Klassifikationsverfahrens unter Benutzung des in x enthaltenen Merkmalssatzes. Die Klassifikationsgenauigkeit kann mit einer Kreuzvalidierung auf einer genügend großen Trainingsmenge abgeschätzt werden.

MERKMALSGENERIERUNG Die Selektion sucht nach geeigneten Unterräumen des Merkmalsraums $F = \{f_1, \ldots, f_n\}$. Ziel der Merkmalsgenerierung ist es hingegen, den Merkmalsraum durch neue Merkmale zu erweitern. Gegeben ist eine Menge von Merkmalsgeneratoren $G = \{g_1, \ldots, g_k\}$. Jedem Generator g_j ist eine Menge von Datentypen zugeordnet, auf denen er operieren kann. Es gibt sowohl unäre als auch binäre Generatoren, die entweder aus einem oder aus zwei Merkmalen ein neues generieren. Beispiele für unäre Generatoren auf numerischen Attributen sind sin, cos, sign etc. Als binäre Generatoren kommen Rechenoperationen wie Multiplikation, Addition, Potenzen etc. in Frage.

Während der Merkmalsgenerierung wird zunächst zufällig ein Operator g_j ausgewählt. Anschließend werden aus den Merkmalen mit geeignetem Datentyp ebenfalls zufällig die benötigte Anzahl ausgewählt und daraus mit Hilfe von g_j ein neues Merkmal erzeugt.

MUTATION Die Mutation erfolgt analog zur Mutation im Fall von Bitstrings. Das Flippen des i-ten Bits entspricht hier der Aktivierung oder Deaktivierung des i-ten Merkmals.

CROSSOVER Auch der Crossoveroperator entspricht im wesentlichem demjenigen des Bitstring-Beispiels. Der einzige Unterschied ist, dass hier die Individuen nicht zwingend die gleiche Länge haben, da der Merkmalsgenerator

unter Umständen einzelnen Individuen neue Merkmale hinzufügt. Daher befinden sich rechts vom Crossover-Punkt nicht immer bei beiden Individuen gleichviele Merkmale. Dennoch werden die Merkmale rechts dieses Punktes unter den Individuen vertauscht.

Parameter

Programmparameter für den evolutionären Algorithmus zur Merkmalsgenerierung sind die Wahrscheinlichkeiten für das An- oder Abschalten eines Merkmals bei der Mutation, die Anzahl neu generierter Merkmale pro Generation sowie die zu verwendenden Generatoren. Hinzu kommen die allgemeinen Parameter wie Abbruchbedingung, Populationsgröße etc.

RITTHOFF et.al. [20] zeigt, dass mit einem solchen Algorithmus gute Ergebnisse erzielt werden können.

3 DATENGENERIERUNG UND DATENEXTRAKTION

Die Bilder von Schauern, welche die MAGIC-Kamera aufnimmt, sowie die Ergebnisse der Analyse-Software MARS werden mit Hilfe der ROOT-Bibliothek [21] in einem binären Format abgelegt. Um sie in RapidMiner zu verarbeiten, müssen diese Daten in ein RapidMiner-kompatibles Format konvertiert werden. Dazu muss das Format der ROOT-Dateien verstanden und sich mit den Zugriffsmöglichkeiten vertraut gemacht werden. Weiterhin muss die Struktur der Daten erschlossen werden, die von den einzelnen MARS-Prorammen erzeugt werden. Da die zu verabeitenden Daten häufig noch nicht die gesamte MARS-Analysekette durchlaufen haben (s. Kap. 2.2), müssen außerdem die einzelnen Analyse-Programme aufgerufen werden.

In diesem Teil wird zunächst die Programmierschnittstelle (Application Programming Interface – API) von ROOT bezüglich Eingabe und Ausgabe erläutert. In Kapitel 3.3 werden dann Struktur und Aufbau des von MARS erzeugten Datenformats beschrieben. Um die Vorverarbeitung größerer Mengen von Einzeldaten zu Vereinfachen, wurden im Rahmen dieser Arbeit Wrapper-Skripte entwickelt, die die Aufrufe der MARS-Programme automatisieren. Diese Skripte sind in Kapitel 3.5 beschrieben.

Die Konvertierung des binären Formats übernimmt das ebenfalls im Rahmen dieser Arbeit entwickelte Programm hillas2ascii, welches in Kapitel 3.6.1 behandelt wird.

3.1 DIE ROOT-BIBLIOTHEK

ROOT ist ein Objekt-orientiertes Framework zur Lösung von Problemen der hochenergetischen Teilchenphysik. Es wird seit Mitte der 1990er Jahre von einem offenen Entwicklerteam unter einer Open-Source Lizenz am CERN entwickelt. Es ist lauffähig unter allen gängigen Betriebssystemen.

Kernstück von ROOT ist eine Sammlung von Bibliotheken, die Klassen zur Analyse auch und insbesondere sehr großer Datenmengen (im Bereich von Terabytes) zur Verfügung stellt. Es existieren neben vielen anderen Klassen Methoden zur statistischen Auswertung und Visualisierung der Daten, sowie zum Speichern großer Datenmengen. Eine Besonderheit von ROOT stellt sein C++-Interpreter CINT dar. Dadurch kann in der Entwicklungsphase eigener Programme der Entwicklungszyklus verkürzt werden, da das Kompilieren entfällt. Alternativ kann das Programm jedoch auch ohne Veränderungen am Quelltext mit einem gängigen Compiler kompiliert werden.

Neben der Möglichkei mit CINT Skripte aus Dateien einzulesen und zu interpretieren, können an der ROOT Command Line auch direkt über die Tastatur eingegebene C++-Ausdrücke evaluiert werden.

Die Programmierung in ROOT ist generell zustandsbehaftet. An vielen Stellen gibt es das Konzept eines „aktiven“ Objekts, auf dem bestimmte Funktionen implizit agieren. So gibt es stets ein aktuelles *Directory*-Object, in dem neue Objekte angelegt werden. Ebenso gibt es bei der Grafik-Ausgabe eine aktive

Canvas, auf der alle neuen grafischen Elemente angelegt werden. Das jeweils aktive Objekt kann entweder explizit durch den Aufruf einer `Cd()`-Funktion oder implizit durch das Instantiieren bestimmter Objekte erfolgen.

3.2 ÜBERSICHT ÜBER DIE PROGRAMMIERSCHNITTSTELLE VON ROOT

Dieses Kapitel gibt einen Überblick über die API von ROOT bezüglich Ein- und Ausgabe, insoweit sie für die Extraktion der MAGIC-Daten relevant ist. Details findet der interessierte Leser in [21], insbesondere in den Kapiteln *Input/Output* und *Trees*.

ROOT speichert Daten in Dateien mit der Endung `.root`. In der Bibliothek werden sie durch die Klasse `TFile` repräsentiert. `TFile` stellt Methoden zum Öffnen, Schreiben und Schließen von Dateien zur Verfügung. Wird eine Instanz von `TFile` erzeugt, wird sie automatisch zum aktiven Directory.

Innerhalb einer Datei sind die Daten in einer Baumstruktur organisiert. Diese besteht aus einem oder mehreren Bäumen (`TTree`). Ein Baum definiert eine Struktur aus Zweigen (`TBranch`) und Blättern. Die Blätter können dabei als unzusammenhängende primitive Datentypen gesehen werden. Eine andere Sicht erlaubt es aber auch, Zweige als Repräsentation von C++-Klassen zu betrachten. Jeder Baum, Zweig und jedes Blatt hat einen Namen, über den es eindeutig identifiziert werden kann.

Ein Baum fasst mehrere gleichartige Datensätze zusammen, die jeweils die vom Baum vorgegebene Struktur haben. Durch die Funktion `TTree::GetEntry()` wird ein beliebiger Datensatz ausgelesen.

Zuvor muss mit Hilfe der Funktion `TTree::SetBranchAddress()` festgelegt werden, welche Teilbäume ausgelesen und wo sie gespeichert werden. Ein Beispiel ist in Listing 3.1 auf der nächsten Seite gegeben.

Die Funktion `TTree::Draw()` erlaubt, neben dem Zugriff auf einzelne Datensätze, ein Histogramm über die Werte ausgewählter Datensätze zu erstellen oder Listen von Datensätzen, die bestimmte Bedingungen erfüllen, zu extrahieren.

Mit Hilfe der GUI-Klassen `TBrowser` und `TTreeViewer` (Abb. 9 auf Seite 28) kann die Struktur einer Datei visualisiert werden. Einzelne Datensätze anzuzeigen ist nicht möglich, jedoch können die Werte eines Blattes über alle Datensätze in ein Histogramm eingetragen oder gegeneinander aufgetragen werden.

3.3 STRUKTUR DER MARS-DATEN

Jedes der MARS-Programme erzeugt Ausgabedateien, deren Strukturen sich voneinander unterscheiden. Im Rahmen dieser Arbeit ist lediglich die Ausgabe von Ganymed relevant. In weiterführenden Arbeiten könnten auch die Daten nach Callisto interessant sein. Die von Star erzeugten Dateien stellen nur einen Zwischenschritt dar und enthält eine Teilmenge der von Ganymed erzeugten Daten.

```
// Oeffne eine Datei, die Hillas-Parameter enthaelt:
TFile f("20071214_00313576_I_CrabNebula-W0.40+035_E.root");

// Die Datei enthaelt einen Baum namens "Events".
// Auf diesen soll ueber den Zeiger events zugegriffen werden:
TTree *events = (TTree*)f.Get("Events");

// Initialisiere Datencontainer:
MHillas *hillas_par = 0;
MHillasExt *ext_hillas_par = 0;

// Verknuepfe Zweige des Baums mit den Datencontainern:
events->SetBranchAddress("MHillas.", &hillas_par);
events->SetBranchAddress("MHillasExt.", &ext_hillas_par);

// Iteriere ueber alle Datensaetze des Baums...
for (int i = 0; i < events->GetEntries(); ++i)
{
  // ...und uebertrage sie in die von den oben angegebenen
  // Zeigern referenzierten Objekte:
  events->GetEntry(i);

  // Die Daten sind nun ueber hillas_par und
  // ext_hillas_par ansprechbar
  do_something(hillas_par, ext_hillas_par);
}
```

Listing 3.1: Beispiel für das Auslesen von Datensätzen aus `TTree`.

3.4 ORGANISATION DER MARS-DATEN

Im Teleskopbetrieb werden einzelne Events aufgenommen. Eine gewisse Anzahl Events wird in *Runs* zusammengefasst und in eine Datei geschrieben. Mehrere zusammengehörige Runs, also zeitlich zusammenhängende Beobachtungen derselben Quelle, werden in *Sequencen* organisiert. Eine Sequenzdatei listet in menschen- und maschinenlesbarer Form die ihr zugehörigen Runs sowie den zugehörigen Pedestal- und Calibration-Run (s. Kap. 2.2.2) auf. Runs und Sequenzen sind fortlaufend nummeriert; Run- und Sequenznummer identifizieren einen Datensatz eindeutig.

Sequenzen können wiederum zu *Datasets* zusammengefasst werden.

Während Callisto und Star auf Sequenzebene arbeiten, erwartet Ganymed als Eingabe einen Dataset und fasst alle relevanten Events darin in einer einzigen Datei zusammen. Alle von MARS erzeugten Dateien sind so organisiert, dass in ihren Trees Datencontainer aus der MARS-Programmbibliothek abgelegt sind. Die Namen der Zweige entsprechen in der Regel den Klassennamen.

3.4.1 *Callistos Dateistruktur*

Wie bereits zuvor beschrieben, kalibriert Callisto die Eingabedaten und schreibt die korrigierten Intensitätswerte und Ankunftszeiten jedes einzelnen Pixels in

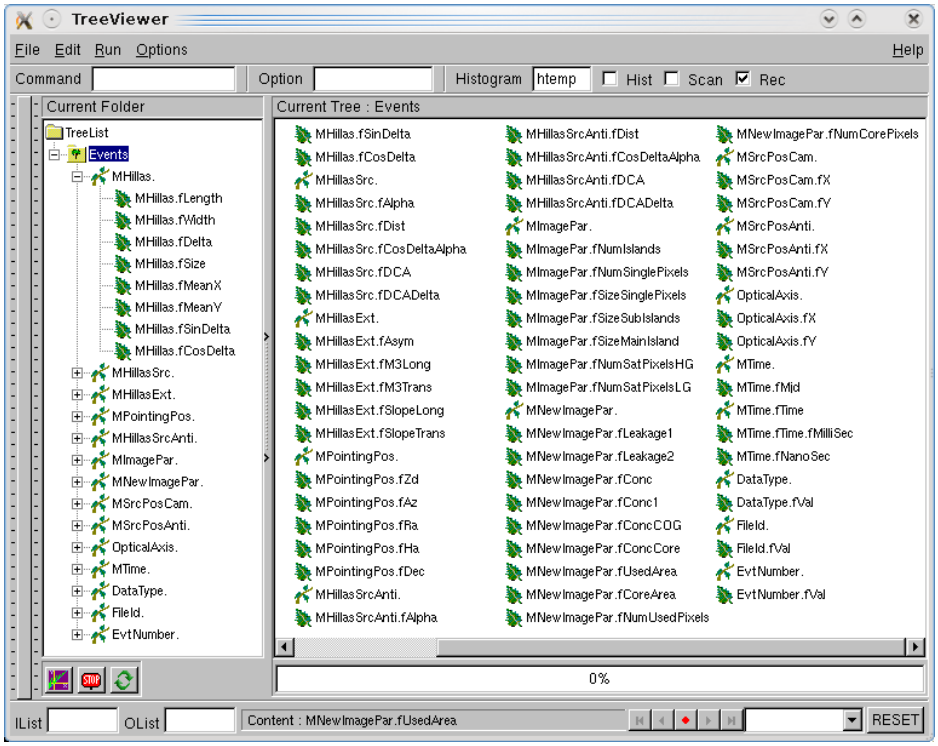


Abbildung 9: Screenshot der Klasse `TTreeViewer`, der den „Events"-Baum einer von Ganymed erzeugten Datei darstellt.

die Ausgabedatei. Diese Daten sind im Baum `Events` im Zweig `MSignalCam` abgelegt. Dieser Baum enthält weiterhin Informationen über den Zeitpunkt der Events (`MTime`) sowie einige weitere Informationen. Falls es sich bei den Daten um Monte-Carlo-generierte Events handelt, werden Informationen dazu in `MMcEvt` abgelegt.

Der Baum `RunHeaders` enthält in `MRawRunHeaders` Informationen über die Anzahl der aufgenommenen Events, Start- und Endzeitpunkt der Beobachtung etc., sowie in `MGeomCam` die Beschreibung Kamerageometrie. Weitere Bäume enthalten Informationen über die Kameraeigenschaften (`Camera`) und weitere Meta-Informationen.

3.4.2 *Ganymeds Dateistruktur*

Ganymed erzeugt zwei verschiedene Dateien: ein *Summary-File* mit allen Events, die die Quality-Cuts überstehen, sowie eine Datei, die nur die als „Gamma" klassifizierten Ereignisse enthält. Die Struktur beider Dateien ist weitgehend identisch.

Nach der Ausführung von Star wird von den Pixelwerten abstrahiert; stattdessen werden die Ereignisse durch die oben beschriebenen Parameter charakterisiert. Diese liegen in der Datei im Baum `Events` in den Zweigen `MHillas`, `MHillasExt`, `MImagePar` und `MNewImagePar` (s. Abb. 9). Quellpositionsbezogene Parameter liegen in `MHillasSrc` bzw. `MHillasAntiSrc`. Da nicht nur eine Anti-Source-Positionen, sondern drei verwendet werden (s. Kap. 2.1.2), enthalten die

Ganymed-Dateien jedes Event viermal, jeweils für unterschiedliche Anti-Source-Positionen.

Die Parameter Theta und Disp liegen in eigenen Zweigen (`ThetaSquared` und `Disp`). Der Zweig `DataType` enthält Informationen über die Semantik des Events: ist der Wert 1, wurden die Parameter bezüglich der echten Quellposition berechnet, ist er 0, bezüglich einer Anti-Source-Position. `PointingPos` gibt an, auf welchen Punkt des Himmels das Teleskop zum Zeitpunkt des Events gerichtet war.

Wird mit Monte-Carlo-generierten Daten gearbeitet, enthält der Zweig `MMcEvt` die Parameter des jeweiligen Events wie Energie, Impuls und Art des Teilchens.

3.5 AUTOMATISIERUNG DER ANALYSE MIT MARS

Die zu analysierenden Daten liegen meist in Form von Callisto-Dateien vor. Als Trainingsdaten benötigt man jedoch Ganymed-Summary-Files. Diese müssen also generiert werden. Unter Umständen sollen sehr viele Sequenzen anaylsiert werden. Um eine Summary-File zu generieren, muss Star auf jeder einzelnen Sequenz aufgerufen werden. Anschließend muss eine Beschreibung des Datasets, der alle Sequenzen enthält, als Eingabe für Ganymed erstellt werden. Zwar besteht bereits eine Datenbank-gestützte Automatisierung dieser Schritte [8], jedoch ist der Zugriff auf die Datenbank nicht immer möglich. Die im Rahmen dieser Arbeit entwickelten Ruby-Skripte [2] automatisieren diesen Prozess ohne die Verwendung der Datenbank.

Alle Skripte lesen die Umgebungsvariablen `ROOTSYS`, `MARSSYS` und `MAGIC_DATA_DIR`, die den Pfad zur ROOT- und MARS-Installation und zum Wurzelpfad der MAGIC-Daten angeben.

3.5.1 *callisto2star und run_star*

run_star ist ein Wrapper-Skript für Star. Als Eingabe erwartet es den Pfad zu den Callisto-Daten, ein Verzeichnis, welches die Sequenzdaten enthält und ein Ausgabeverzeichnis sowie die Konfigurationdatei für Star. run_star ruft dann Star für jede gefundene Sequenz auf.

Durch den Aufruf `ruby run_star.rb -help` werden alle verfügbaren Optionen angezeigt.

Das Skript callisto2star wiederum ist ein Wrapper für run_star, der mehrere Tupel von Eingabe-, Ausgabe- und Sequenzverzeichnis in einer Konfigurationsdatei verwaltet. Listing 3.2 zeigt ein Beispiel für eine solche Konfigurationdatei.

Der Aufruf von callisto2star hat die Form:

```
ruby callisto2star.rb <configfile> [<cfg\_set> [<cfg\_set>
    [...]]]
```

Wird kein Konfigurations-Set angegeben, werden alle Sets aus der Konfigurationsdatei verarbeitet.

```
$sets[:gamma_mc] = {
  :callisto_dir => "#{$datadir}/path/to/gamma/callisto/data",
  :sequence_dir => "#{$datadir}/path/to/gamma/sequence/files",
  :star_dir => "#{$datadir}/path/where/to/place/gamma/star/
      files",
  :mc => true
}

$sets[:hadronen_real] = {
  :callisto_dir => "#{$datadir}/path/to/hadron/callisto/data",
  :sequence_dir => "#{$datadir}/path/to/hadron/sequence/files",
  :star_dir => "#{$datadir}/path/where/to/place/hadron/star/
      files",
  :mc => false
}
```

Listing 3.2: Beispiel für eine Konfigurationsdatei von callisto2star.

3.5.2 *star2ganymed und run_ganymed*

run_ganymed und star2ganymed funktionieren analog zu run_star und callisto2star. run_ganymed fasst alle gefundenen Sequenzen zu einem Dataset zusammen und ruft Ganymed auf diesem auf. star2ganymed liest eine Konfigurationsdatei ein, die strukturell den gleichen Aufbau wie die von callisto2star hat. Auch der Aufruf ist analog:

```
ruby star2ganymed.rb <configfile> [<cfg_set> [<cfg_set> [...]]]
```

star2ganymed erzeugt sowohl die Standardausgabedatei von Ganymed wie auch eine Summary-File für jeden Dataset.

3.6 EXTRAKTION VON DATEN IM ROOT-FORMAT

3.6.1 *hillas2ascii*

Das Programm hillas2ascii dient dazu, aus den Ganymed Summary-Files die Hillas-Parameter und die übrigen Bildparameter in ein ASCII-Format zu extrahieren. Es können wahlweise der Inhalt einer oder mehrerer kompletter Dateien extrahiert werden, oder aber bestimmte Teilmengen aus den Dateien.

Die Datenverarbeitung erfolgt schrittweise (s. Abb. 10). Zunächst werden die Eingangsdaten, die Ganymed-Dateien, von der Klasse `InputFile` gelesen. `InputFile` verwaltet interne Listen der Ereignisse, die im letzten Schritt vom `EventWriter` in die Ausgabedatei geschrieben werden. Zunächst enthalten diese Listen alle Events.

Im nächsten Schritt werden Schnitte angewendet, d.h. Events, die bestimmte Bedingungen nicht erfüllen, werden aus den Ereignislisten entfernt. Hier können zum Beispiel alle Events, deren Parameter bezüglich der Anti-Source-Position berechnet wurden, aussortiert werden.

Eine weitere Vorverarbeitung leistet die Klasse `BinningBalancer`, die hilfreich ist, wenn die Eingangsdaten unterschiedliche Event-Klasen enthalten und sichergestellt werden soll, dass die Verteilung einer bestimmten Größe (wie

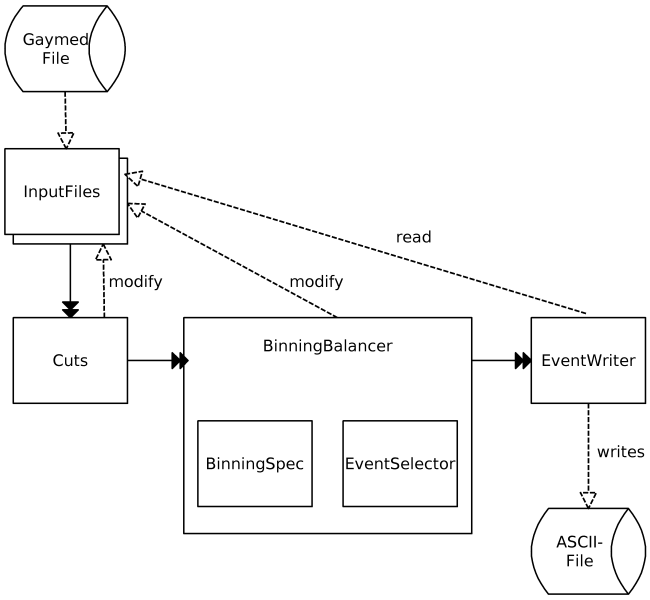


Abbildung 10: Verarbeitungsschritte in hillas2ascii

zum Beispiel dem Zenitwinkel) in beiden Klassen identisch ist. Als Eingabe erhält diese Klasse neben den Eventlisten die Anzahl der Gamma- und Hadron-Ereignisse, die extrahiert werden soll. Dann ordnet er die Events getrennt nach Event-Klassen in ein Histogramm ein und entfernt die Bins, in denen das Verhältnis zwischen Gamma- und Hadron-Events nicht ausgeglichen ist. Sobald alle Events in solchen Bins entfernt wurden, wird aus den übrigen mit Hilfe der Klasse `EventSelector` eine gewisse Menge von Events ausgewählt. In dieser Arbeit wurde ein `EventSelector` verwendet, der die Auswahl zufällig vornimmt.

Die Klasse `BinningSpec` definiert einige Vorgaben für das Binning, wie die Anzahl der Bins.

Nach diesen Vorverarbeitungsschritten liest der `EventWriter` die verbleibenden Events aus den Listen aus und schreibt sie in eine Komma-separierte ASCII-Datei. In dieser Arbeit wurde die Klasse `HillasEventWriter` verwendet, die gerade die gewünschten Bildparameter schreibt. Die Ausgabe kann ohne weitere Modifikationen in RapidMiner verwendet werden.

Die verwendeten Ganymed-Dateien, Cuts und Binning-Spezifikationen werden über eine Konfigurationsdatei gesteuert Listing 3.3) zeigt eine Konfiguration zur Extraktion von 10000 Gamma- und 50000 hadronischen Events, deren Zenitwinkelverteilungen identisch sind. Die Syntax der Konfigurationsdatei folgt der YAML-Spezifikation (s. [6]).

Erweiterungen

Die Struktur des Programms ist so beschaffen, dass Modifikationen und Erweiterungen leicht vorgenommen werden können. Durch Ableiten der Klasse `EventWriter` könnten beispielsweise anstelle der Hillas-Parameter die Intensitätswerte nach Callisto ausgegeben werden.

```
inputfiles:
  - name: /path/to/ganymed/hadron/summary/file.root
    label: Hadron
  - name: /path/to/ganymed/gamma/summary/file.root
    label: Gamma
cuts:
  - condition: DataType.fVal == 1
    label: Gamma
  - condition: DataType.fVal == 0
    label: Hadron
binningbalancer:
  binningspec:
    quantity: MPointingPos.fZd
    numbins: 50
    minvalue: 0
    maxvalue: 50
  numg: 10000
  numh: 50000
  eventselector:
    seed: 2009
eventwriter:
  filename: test.csv
```

Listing 3.3: Konfigurationsdatei für hillas2ascii. Die Cuts entfernen alle Events, die auf die falsche Quellposition bezogen sind (Anti-Source bei Gammas, Source bei Hadronen). Im Abschnitt `binningbalancer` sind Parameter für das Binning, die Anzahl der auszuwählenden Events sowie der Seed für den zufallsbasierten Auswahloperator gegeben. Die Konfiguration des `eventwriter` gibt den Namen der Ausgabedatei an.

4 EXPERIMENTE

In den Experimenten werden die in Teil 2 beschriebenen Verfahren angewendet, um Merkmale zu finden, die Gammas und Hadronen besser charakterisieren. Die gefundenen Merkmale werden anschließend validiert, d.h. daraufhin überprüft, ob sie generalisierbar sind und nicht nur auf der Trainingsmenge eine gute Klassifikationsleistung ermöglichen. Diese Klassifikationsleistung wird zuletzt mit der Leistung des mit Ganymed bestehenden Verfahrens verglichen.

Alle Experimente werden mit Hilfe von RapidMiner durchgeführt.

4.1 VORAUSSETZUNGEN

4.1.1 *Anforderungen an die Daten*

Voraussetzung für alle Experimente ist zunächst das Vorhandensein geeigneter Daten. Die Daten des MAGIC-Teleskops werden im wesentlichen durch drei Parameter charakterisiert: dem *Zenitwinkel*, der *Point Spread Function* (PSF) sowie der beobachteten Quelle.

Der Zenitwinkel gibt an, unter welchem Beobachtungswinkel die Daten aufgenommen wurden: der Zenitwinkel gibt den Winkel zwischen der vertikalen, also dem Zenith, und der Beobachtungsachse an. Je größer der Zenitwinkel ist, desto näher liegt die Quelle am Horizont. Ein veränderter Zenitwinkel bedeutet, dass das beobachtete Cherenkov-Licht eine unterschiedlich lange Strecke in der Erdathmosphäre zurücklegt, bevor es auf das Teleskop trifft. Dadurch entstehen Veränderungen in den aufgenommenen Daten.

Die PSF ist eine Größe, die sich aus den Beobachtungsbedingungen wie Eigenschaften der Atmospähre, Leistung des Teleskops etc. ergibt, und meist über größere Zeiträume konstant ist.

Ein Verfahren zur Klassifikation der beobachteten Ereignisse wird nur dann gute Ergebnisse liefern, wenn diese drei Parameter zwischen Trainingsmenge und den unbekannten Beispielen in etwa übereinstimmt. Außerdem müssen sie in der Trainingsmenge bei positiven und negativen Beispielen übereinstimmen.

4.1.2 *Verwendete Daten*

In dieser Arbeit werden zwei Datensätze verwendet: die Beispiele für Gamma-Events wurden durch den CORSIKA-Simulator erzeugt[1] und weisen eine PSF von $13,1$ mm sowie eine Zenitwinkelverteilung von 0° bis 48° auf. Die Beispiele für Hadronen wurden aus echten Aufnahmen des Krebsnebels im Wobble-Modus extrahiert[2]. Diese Aufnahmen enthalten Zenitwinkel von 15° bis 48° bei einer PSF von $12,9$ mm. Beide Datensätze enthalten jeweils etwa $1 \cdot 10^6$ Events. Zur Merkmalsgenerierung und -Selektion wurde eine Menge S_t von

1 Es handelt sich um die Sequenzen 6842 bis 8339, näheres s. [1]

2 Sequenz 313528, s. [1]

1000 Gamma-Events und 5000 hadronische Events zufällig ausgwählt, deren Zenitwinkel gleichverteilt zwischen 15 ° und 48 ° sind.

Der Überhang an negativen Beispielen soll das Experiment etwas näher an die Realität heranführen, in der auf ein Gamma-Event etwa 100 bis 1000 hadronische Events kommen.

Zur Validierung der Ergebnisse wird eine weitere Teilmenge S_v der Daten gewählt. Da der Rechenaufwand im Vergleich zur Merkmalsgenerierung und -Selektion relativ gering ist, ist die Testmenge zehnmal größer als die Trainingsmenge. Da die Ergebnisse beider Mengen zufällig gewählt und die Datenmenge sehr groß ist, ist die Schnittmenge von S_v und S_t sehr klein, was wichtig für die Aussagekraft der Validierung ist.

4.2 MERKMALSGENERIERUNG UND -SELEKTION

4.2.1 *Grundlegender Aufbau des Experiments*

Zur Merkmalsgenerierung und -Selektion wird ein RapidMiner-Prozess wie in Abbildung 11 verwendet. Kernstück dieses Prozesses ist der FeatureSelection-Generation-Operator vom Typ YAGGA2. Dieser Operator implementiert einen evolutionären Algorithmus zur Merkmalsgenerierung (s. Kap. 2.3.6). In jeder Generation generiert er neue Merkmale und aktiviert und deaktiviert vorhandene Merkmale für jedes Individuum der Population. Die neuen Merkmalssätze werden mit Hilfe seines inneren Operators, der Kreuzvalidierung (XValidation-Parallel), bewertet. Wie in Kapitel 2.3.4 beschrieben teilt die Kreuzvalidierung die Beispielmenge mehrmals in disjunkte Teilmengen auf, trainiert ein Modell, in diesem Fall eine SVM, auf einem Teil der Daten und evaluiert sie mit Hilfe des ModelApplier auf den übrigen. Sie liefert die mittlere Klassifikationsgenauigkeit über allen Teilmengen. Diese dient dem evolutionärem Algorithmus dazu, die aktuelle Merkmalsmenge zu bewerten.

Die übrigen Operatoren dienen lediglich dem Logging und zur Darstellung von Zwischenergebnissen. Unter anderem werden die verwendeten Merkmale und Konstruktionsbeschreibungen für die generierten Merkmale in eine Datei geschrieben, so dass sie in weiteren Prozessen verwendet werden können.

4.2.2 *Konfiguration des Prozesses*

Der in Abbildung 11 gezeigte Aufbau beinhaltet als Klassifikator eine Support Vector Machine. Anstelle dieses Klassifikators wurde auch k-Nearest-Neighbors mit $k = 20$ verwendet. Auch wurde anstelle des linearen Kerns eine radiale Kernelfunktion für die SVM eingesetzt.

Der evolutionäre Algorithmus wurde in allen Experimenten mit einer Populationsgröße von 10 und einer maximalen Anzahl von 30 Generationen konfiguriert. Zur Generierung neuer Merkmale wurden die vier Grundrechenarten, Potenzen, e-Funktion, Logarithmus, Quadratwurzel, Kehrwertbildung, Sinus, Cosinus, Tangens, Arcustanges sowie Betragsbildung zugelassen.

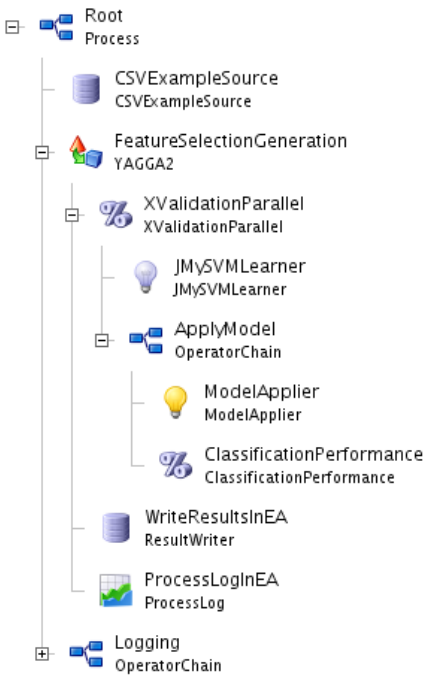


Abbildung 11: RapidMiner-Prozess zur Merkmalsgenerierung und -Selektion

4.3 VALIDIERUNG DER GENERIERTEN PARAMETER

Zur Validierung der Merkmalssätze wird ein Aufbau nach Abbildung 12 auf der nächsten Seite verwendet. Zunächst wird der Beispieldatensatz eingelesen. Wie zuvor beschrieben sollte dieser Datensatz eine möglichst kleine Schnittemenge mit dem zur Generierung verwendeten Datensatz haben.

In der Preprocessing-OperatorChain werden der im vorherigen Prozess generierte Merkmalssatz und und die Attribut-Gewichte wiederhergestellt. Eine Kreuzvalidierung übernimmt dann die eigentliche Bewertung der Merkmale. Wie schon bei der Merkmalsgenerierung werden auch zur Validierung unterschiedliche Klassifikatoren innerhalb der Kreuzvalidierung eingesetzt.

Die letzten beiden Operatoren dienen wieder dem Schreiben der Ergebnisse; insbesondere wird bei Verwendung der SVM die Definition der trennenden Hyperebene gespeichert, um sie in weiteren Prozessen ohne erneutes Training zur Klassifikation verwenden zu können.

Eine Alternative zu diesem zweiten Aufbau scheint zunächst eine Kreuzvalidierung im ersten Aufbau zu sein, die den EA beinhaltet. Dadurch würden jedoch nicht die generierten Merkmalssätze validiert, sondern das Verfahren des EA an sich: aufgrund der Zufallsprozesse im EA würde wahrscheinlich in den einzelnen Iterationen nicht derselbe Merkmalssatz generiert werden, so dass keine Aussage über einen speziellen Merkmalssatz getroffen werden kann. Zudem ist es aufgrund des hohen Rechenaufwandes eher ineffizient, einen EA zur Merkmalsgenerierung innerhalb einer Kreuzvalidierung zu verwenden.

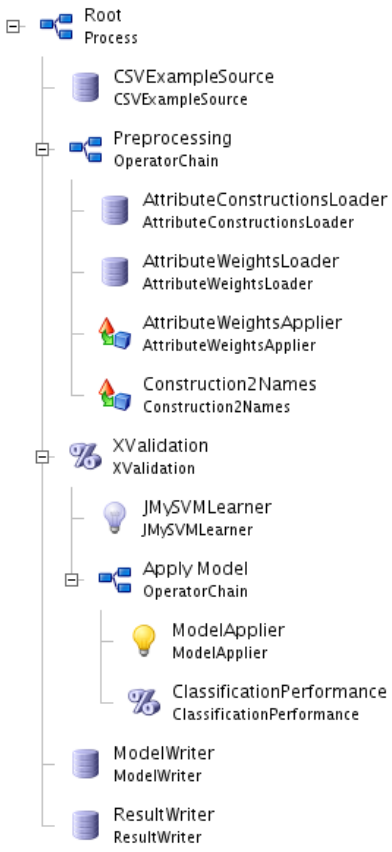


Abbildung 12: RapidMiner-Prozess zur Merkmalsvalidierung

4.4 VERGLEICH MIT GANYMED

Um die erreichte Klassifikationsgüte einordnen zu können, muss sie mit der Leistung des vorhandenen Klassifikationsverfahren, d.h. mit der Leistung von Ganymed, verglichen werden. Von den in dieser Arbeit verwendeten Gütemaßen kann in Ganymed lediglich der Gamma-Recall berechnet werden. Hierzu eignen sich die Monte-Carlo-generierten Gamma-Daten. Die Klassifikationsgüte bzgl. des Untergrundes kann nicht angegeben werden: die Aufnahmen des Krebsnebels im Wobble-Modus sind ungeeignet, da Ganymed stets die On-Positionsbasierten Bildparameter zur Klassifikation nutzt. Während für die Off-Positionsbasierten Daten das Label bekannt ist (Hadron), ist dies für die Parameter der echten Quellposition nicht der Fall; hier können sowohl Gamma- als auch hadronische Events auftreten.

Zum Vergleich mit Ganymed wird also überprüft, wie viele der simulierten Gamma-Events RapidMiner mit dem generierten Merkmalssatz und Ganymed jeweils korrekt als Gamma klassifizieren. Als Testmenge wird hier der gesamte Datensatz der Gamma-Events (s. Kap. 4.1.2) verwendet. Er beinhaltet etwa 1 Mio. Beispiele. Bei Verwendung der SVM wird die im vorherigen Prozess gelernte trennende Hyperebene verwendet.

5 ERGEBNISSE UND INTERPRETATION

5.1 GENERIERTE MERKMALSSÄTZE

Im folgenden sind die generierten Merkmalssätze zusammen mit den im evolutionären Algorithmus verwendeten Klassifikatoren aufgelistet. Die damit erzielten Klassifikationsleistungen sind im nächsten Abschnitt angegeben.

Merkmal	**Konstruktion**
ImageParNew_Leakage2	ImageParNew_Leakage2
gensym34	(ImagePar_NumSinglePixels + (ImageParNew_Leakage2 / ImagePar_SizeSinglePixels))
gensym8	(ImageParNew_Leakage2 / ImagePar_SizeSinglePixels)
ImagePar_NumSinglePixels	ImagePar_NumSinglePixels
ImagePar_SizeSubIslands	ImagePar_SizeSubIslands
Hillas_Width	Hillas_Width

Tabelle 2: Erster Merkmalssatz mit kNN, k = 20

Merkmal	**Konstruktion**
Hillas_Width	Hillas_Width
ImageParNew_Conc1	ImageParNew_Conc1
ImagePar_NumSinglePixels	ImagePar_NumSinglePixels
ImageParNew_Leakage2	ImageParNew_Leakage2
gensym8	(ImageParNew_Leakage2 / ImagePar_SizeSinglePixels)
ImagePar_SizeSubIslands	ImagePar_SizeSubIslands
gensym25	log(ImageParNew_Conc1)
gensym34	(ImagePar_NumSinglePixels + (ImageParNew_Leakage2 / ImagePar_SizeSinglePixels))

Tabelle 3: Zweiter Merkmalssatz mit kNN, k = 20

Merkmal	**Konstruktion**
Hillas_CosDelta	Hillas_CosDelta
Hillas_SinDelta	Hillas_SinDelta
HillasExt_M3Long	HillasExt_M3Long
ImagePar_SizeSubIslands	ImagePar_SizeSubIslands
ImagePar_SizeMainIsland	ImagePar_SizeMainIsland
ImageParNew_Leakage2	ImageParNew_Leakage2
ImageParNew_ConcCOG	ImageParNew_ConcCOG
ImageParNew_NumUsedPixels	ImageParNew_NumUsedPixels
Hillas_MeanY	Hillas_MeanY
Hillas_Area	Hillas_Area
HillasExt_Asym	HillasExt_Asym
ImagePar_NumSinglePixels	ImagePar_NumSinglePixels
ImagePar_SizeSinglePixels	ImagePar_SizeSinglePixels
gensym22	const[-6.7379864143703 43]()

Tabelle 4: Merkmalssatz mit SVM (lineare Kernelfunktion)

Merkmal	**Konstruktion**
Hillas_Width	Hillas_Width
ImagePar_SizeSinglePixels	ImagePar_SizeSinglePixels
ImageParNew_NumCorePixels	ImageParNew_NumCorePixels
gensym10	(ImageParNew_UsedArea - ImagePar_SizeSinglePixels)
gensym4	(Hillas_Width / Hillas_Area)
ImagePar_NumSinglePixels	ImagePar_NumSinglePixels

Tabelle 5: Merkmalssatz mit SVM (radiale Kernelfunktion)

5.2 PERFORMANZ DER GENERIERTEN MERKMALSSÄTZE

Tabelle 6 listet die Klassifikationsleistung der jeweiligen Merkmalssätze in Form der Accuracy auf. Jeder Merkmalssatz wurde, wenn möglich, mit kNN ($k = 20$), einer SVM mit linearer und einer SVM mit radialer Kernfunktion validiert. Die Testmenge bestand aus 10000 Gamma-Events und 50000 hadronischen Events.

Die erste Spalte gibt an, mit welchem Klassifikator der jeweilige Merkmalssatz generiert wurde (s. oben). Die mit *Training* betitelte Spalte enthält die Accuracy, die der Merkmalssatz auf der Trainingsmenge des EA erreicht hat. Die übrigen Spalten geben die Accuracy des Merkmalssatzes mit dem jeweils in der Spalte angegebenen Klassifikators auf der Testmenge.

Merkmalssatz	**Training**	**kNN**	**SVM lin.**	**SVM rad.**
kNN (1)	$94.80\% \pm 0.75\%$	$94.85\% \pm 0.29\%$	n/a	n/a
kNN (2)	$94.76\% \pm 0.75\%$	$94.84\% \pm 0.30\%$	n/a	n/a
SVM rad.	$95.77\% \pm 0.63\%$	$91.63\% \pm 0.37\%$	$95.10 \pm 0.22\%$	$84.35\% \pm 23.68\%$
SVM lin.	$95.40\% \pm 0.69\%$	$90.27\% \pm 0.35\%$	$95.86\% \pm 0.19\%$	$95.34\% \pm 0.16\%$

Tabelle 6: Accuracy der linearen SVM, der SVM mit RBF als Kernfunktion und des kNN-Lerners auf der Testmenge unterschiedlicher Klassifikatoren auf den generierten Merkmalssätzen. Die Spalte Training gibt die innerhalb des evolutionären Algorithmus' erzielte Accuracy an.

Bei fast allen Experimenten liegt der Recall der Gamma-Events zwischen etwa 80 % und 85 %, bei den hadronischen Events zwischen 95 % und 99 %. Die Precision der Gamma-Klasse liegt nur zwischen 70 % und 95 %, wobei der kNN-Klassifikator bei der Validierung die schlechtesten Ergebnisse liefert. Die Performanz der linearen und der radialen SVM unterscheiden sich kaum. In Anhang A.1 sind Recall und Precision sowie die Gewichtung der Attribute für alle Experimente aufgeführt.

Offensichtlich liefert der Merkmalssatz, der mit der linearen SVM generiert wurde, insgesamt die beste Accuracy, vor allem dann, wenn darauf erneut eine lineare SVM angewendet wird. Daher wird dieser Merkmalssatz mit linearer SVM auf einer Testmenge aus allen verfügbaren Beispielen angewendet. Dabei konnte eine Accuracy von 94,96% erzielt werden. Tabelle 7 zeigt Precision und Recall:

5.3 VERGLEICH MIT GANYMED

Wie in Kapitel 4.4 beschrieben, kann die Performanz von Ganymed lediglich auf den Monte-Carlo-Daten ausgewertet werden. Von den etwa 1,1 Mio. Ereignissen stuft Ganymed nur etwa 238.000 als Gamma ein. Das entspricht einem Recall von nur 21 %. Allerdings wurden die in Ganymed verwendeten Heuristiken nicht hinsichtlich der Accuracy optimiert, sondern hinsichtlich der Signifikanz nach [16]. Aufgrund dieser Optimierung nach unterschiedlichen

True	Hadron	Gamma	Precision
Hadron	3845460	189521	95,07%
Gamma	64297	932540	93,11%
Recall	98,33%	79,68%	
Accuracy	94,96%		

Tabelle 7: Precision und Recall bei der Klassifikation aller verfügbaren Beispiele mit einer linearen SVM und dem mit Hilfe der linearen SVM generierten Merkmalssatz.

Gütemaßen, und da in dieser Arbeit das Gamma-Hadron-Verhältnis nicht berücksichtigt wurde, sind jedwede Vergleiche nicht aussagekräftig. In weiteren Arbeiten sollte physikalisches Vorwissen bereits bei der Merkmalsgenerierung berücksichtigt werden.

5.4 BEWERTUNG

Die obige Tabelle 7 zeigt, dass auf der Testmenge bei einer Accuracy von 95% über 98% der hadronischen Events ausgefiltert werden konnten. Allerdings wurden nur knapp 80% der Gamma-Events korrekt erkannt. Dies scheint ein gutes Ergebnis zu sein.

Bedenkt man jedoch, dass bei realen Beobachtungen auf ein Gamma etwa 1000 Hadronen kommen, müssen die Hadronen ein stärkeres Gewicht erhalten. Bei Wahl von $f = 287$ ergibt sich nach Gl. (29) und (30) Tabelle 8. Der Faktor $f = 287$ wurde gewählt, da das Verhältnis zwischen Gammas und Hadronen bei echten Daten bei $\frac{1}{1000}$ liegt, während es in der Testmenge bei $\frac{1}{3{,}48}$ liegt.

True	Hadron	Gamma	Precision
Hadron	3845460	189521	99,98%
Gamma	64297	932540	4,81%
Recall	98,33%	79,68%	
Accuracy	98,34%		

Tabelle 8: Precision und Recall bei der Klassifikation aller verfügbaren Beispiele mit einer linearen SVM und dem mit Hilfe der linearen SVM generierten Merkmalssatz, bei einer Gewichtung der hadronischen Events mit $f = 287$.

In fortführenden Experimenten sollte dieser Faktor unbedingt bereits bei der Generierung des Merkmalssatzes berücksichtigt werden. Ohne dessen Berücksichtigung führt die Merkmalsselektion offenbar zu keinen zufriedenstellenden Ergebnissen.

Bemerkenswert ist jedoch, dass der beste Merkmalssatz fast ausschließlich durch Merkmalsselektion entstanden ist (vgl. Tab. 4). Ohne Berücksichtigung

des Gamma-Hadron-Verhältnisses scheinen die vorhandenen Bildparameter also gut geeignet zu sein, die Teilchen-Klassen zu beschreiben. Möglicherweise könnten jedoch weitere, bessere Merkmale generiert werden, wenn der EA dahingehend modifiziert wird, dass er einen größeren Fokus auf die Merkmalsgenerierung setzt und pro Generation und Individuum mehr Merkmale erzeugt werden.

Eine Verbesserung kann unter Umständen auch erzielt werden, wenn anstelle der Anti-Source-Daten aus dem Wobble-Mode echte Off-Daten verwendet werden, denn die Anti-Source-Daten enthalten in Wirklichkeit nicht nur hadronische Events, sondern auch einen gewissen Anteil an Gamma-Events.

Der größte Sprung in der Verbesserung des Qualifikationsergebnisses wird sich aber vermutlich ergeben, wenn das Gamma-Hadron-Verhältnis bereits im evolutionären Algorithmus bei der Bewertung der Merkmalsmenge berücksichtigt wird. Ebenfalls sollten in der SVM die Slackvariablen der Hadron-Beispiele entsprechend gewichtet werden.

5.5 AUSBLICK

Diese Arbeit untersucht die Generierung neuer Merkmale auf Grundlage der Hillas-Parameter und weiterer bekannter Merkmale. Vorwissen über die Beziehungen zwischen den Merkmalen oder Berechnungen, die Ganymed auf diesen Merkmalen durchführt, werden nicht genutzt. Zwar wird der evolutionäre Algorithmus dieses Wissen bei ausreichend langer Laufzeit vielleicht entdecken; Vorverarbeitungsschritte, die die Kenntnis physikalischer Grundlagen ausnutzen, könnten die Performanz jedoch bei gleicher Laufzeit steigern.

Ein weiteres interessantes Forschungsgebiet tut sich auf, wenn diese Beziehungen für das Training von Klassifikatoren ohne negative Beispiele, also nur auf Basis von Monte-Carlo-generierten Gamma-Events, genutzt wird. Möglicherweise bestehen zwischen einigen Merkmalen Relationen, die die Vorhersage eines weiteren Merkmals ermöglicht. Durch Vergleich des vorhergesagten Wertes mit dem tatsächlichen, aus dem Bild errechneten Wert kann unter Umständen eine Klassifikation erfolgen.

Überhaupt ergibt sich die Frage, ob die Klassifikation der Aufgabe angemessen ist. Vielleicht können die Gamma-Strahlen besser als interessante Ausreißer erkannt werden. Auch die neuen Varianten der SVM zur Datenbeschreibung bzw. die Lösung des Einklassenproblems (Gammastrahlen ermitteln, alles andere ausgrenzen) mit Hilfe von verschiedenen Kernfunktionen sind eventuell der Aufgabe angemessener.

Alle bisherigen Verfahren zur Gamma-Hadron-Separation ruhen auf dem Fundament der Hillas-Parameter. Diese wurden bereits 1985 von Hillas (vgl. [15]) vorgeschlagen. Zwar wurden immer wieder verbesserte Parameter eingeführt, die jedoch die grundsätzliche Eignung der Hillas-Paramter nicht in Frage stellen. Möglicherweise erlauben es die heutigen Verfahren des Data Mining und die gesteigerte Rechenleistung moderner Computer jedoch, direkt aus den Rohdaten neue, noch besser geeignete Merkmale zu extrahieren.

Auch durch die Inbetriebnahme des Schwester-Teleskops MAGIC II ergeben sich neue Forschungsgebiete. Durch die Verwendung des zweiten Teleskops kann sich aufgrund der nun möglichen stereoskopischen Beobachtungen zwar

eine Steigerung der Sensitivität ergeben, jedoch nur, wenn die zusätzlichen Daten auf geeignete Art und Weise genutzt und analysiert werden.

Somit besteht auch über den Ansatz dieser Arbeit hinaus weiterer Forschungsbedarf auf dem Gebiet der Gamma-Hadron-Separation im MAGIC-Experiment.

A ANHANG

A.1 DETAILLIERTE AUFLISTUNG ALLER ERGEBNISSE

Die folgenden Tabellen listen Recall und Precision aller angewendeten Klassifikatoren auf der Testmenge auf.

A.1.1 *Klassifikationgüte*

k-Nearest-Neighbors (1)

Performance des ersten mit Hilfe von kNN erzeugten Merkmalssatz.
Accuracy:

True	**Gamma**	**Hadron**	**Precision**
Gamma	8041	1101	86,31%
Hadron	1991	48927	95,93%
Recall	75,24%	97,75%	
Accuracy	$94,85\% \pm 0,29\%$		

Tabelle 9: Training: kNN-20 (1), Validierung: kNN-20

k-Nearest-Neighbors (2)

Performance des zweiten mit Hilfe von kNN erzeugten Merkmalssatz.

True	**Gamma**	**Hadron**	**Precision**
Gamma	8094	1161	85,66%
Hadron	1938	48867	96,03%
Recall	76,06%	97,62%	
Accuracy	$94,84\% \pm 0,30\%$		

Tabelle 10: Training: kNN-20 (2), Validierung: kNN-20

Lineare SVM

Performance des mit Hilfe der linearen SVM erzeugten Merkmalssatz.

true	**Gamma**	**Hadron**	**Precision**
Gamma	5398	1207	77,64%
Hadron	4634	48821	90,51%
Recall	14,15%	97,53%	
Accuracy	$90,27\% \pm 0,35\%$		

Tabelle 11: Training: lin. SVM, Validierung: kNN-20

true	**Gamma**	**Hadron**	**Precision**
Gamma	8351	804	90,37%
Hadron	1681	49224	96,58%
Recall	79,87%	98,37%	
Accuracy	$95,86\% \pm 0,19\%$		

Tabelle 12: Training: lin. SVM, Validierung: lin. SVM

true	**Gamma**	**Hadron**	**Precision**
Gamma	7789	555	92,87%
Hadron	2243	49473	95,47%
Recall	71,20%	98,88%	
Accuracy	$95,34\% \pm 0,16\%$		

Tabelle 13: Training: lin. SVM, Validierung: rad. SVM

Radiale SVM

Performance des mit Hilfe der radialen SVM erzeugten Merkmalssatz.

true	**Gamma**	**Hadron**	**Precision**
Gamma	6873	1867	72,84%
Hadron	3159	48161	93,44%
Recall	54,04%	96,12%	
Accuracy	$91,63\% \pm 0,37\%$		

Tabelle 14: Training: rad. SVM, Validierung: kNN-20

true	**Gamma**	**Hadron**	**Precision**
Gamma	8007	918	88,54%
Hadron	2025	49110	95,88%
Recall	74,71%	98,13%	
Accuracy	$95,10\% \pm 0,22\%$		

Tabelle 15: Training: rad. SVM, Validierung: lin. SVM

true	**Gamma**	**Hadron**	**Precision**
Gamma	8645	8010	7,35%
Hadron	1387	42018	96,70%
Recall	83,96%	80,94%	
Accuracy	$84,35\% \pm 23,68\%$		

Tabelle 16: Training: rad. SVM, Validierung: rad. SVM

A.1.2 *Attributgewichte*

	Gewicht	
Merkmal	**SVM lin.**	**SVM rad.**
Hillas_MeanY	-0,040	-138,558
Hillas_Area	3,309	2226,636
Hillas_CosDelta	0,007	27,146
Hillas_SinDelta	-0,005	42,149
HillasExt_Asym	0,071	201,062
HillasExt_M3Long	0,111	34,991
ImagePar_NumSinglePixels	-2,911	-2381,703
ImagePar_SizeSinglePixels	1,085	-1198,363
ImagePar_SizeSubIslands	0,633	1690,058
ImagePar_SizeMainIsland	-0,336	-1974,664
ImageParNew_Leakage2	-0,296	-231,377
ImageParNew_ConcCOG	0,273	-724,556
ImageParNew_NumUsedPixels	-0,406	863,921
Const[-6,737986414370343]	0,000	0,000

Tabelle 17: Merkmalsgewichte des mit Hilfe der linearen SVM generierten Merkmalssatzes bei Anwendung unterschiedlicher SVMs auf die Testmenge.

	Gewicht	
Merkmal	**SVM lin.**	**SVM rad.**
Hillas_Width	1,388	336,835
ImagePar_NumSinglePixels	-2,934	-264,209
ImagePar_SizeSinglePixels	1,184	-94,628
ImageParNew_NumCorePixels	0,255	160,758
gensym10	-0,536	154,635
gensym4	-0,053	-124,657

Tabelle 18: Merkmalsgewichte des mit Hilfe der radialen SVM generierten Merkmalssatzes bei Anwendung unterschiedlicher SVMs auf die Testmenge.

A.2 ABKÜRZUNGEN

Abkürzung	Bedeutung
EA	Evolutionärer Algorithmus
IDE	Integrated Development Environmeht
kNN	k-Nearest-Neighbors
MAGIC	Major Atmospheric Gamma-Ray Imaging Cherenkov Telescopes
PMT	Photo Multiplier Tube
PSF	Point Spread Function
SMO	Sequential Minimal Optimization
SVM	Support Vector Machine

Tabelle 19: In dieser Arbeit verwendete Abkürzungen

TABELLENVERZEICHNIS

ABBILDUNGSVERZEICHNIS

LISTINGS

LITERATURVERZEICHNIS

[1] *MAGIC Database Websites*. http://db.astro.uni-wuerzburg.de. Version: 29.07.2009 (Zitiert auf Seite 33.)

[2] *Ruby Programming Language*. http://www.ruby-lang.org. Version: 29.07.2009 (Zitiert auf Seite 29.)

[3] *CORSIKA an Air Shower Simulation Program*. http://www-ik.fzk.de/corsika/. Version: 30.07.2009 (Zitiert auf Seite 3.)

[4] Aliu, E. ; Anderhub, H. ; Antonelli u. a.: Improving the performance of the single-dish Cherenkov telescope MAGIC through the use of signal timing. In: *Astroparticle Physics* 30 (2009), Januar, S. 293–305. http://dx.doi.org/10.1016/j.astropartphys.2008.10.003. – DOI 10.1016/j.astropartphys.2008.10.003 (Zitiert auf Seite 9.)

[5] Backes, M.: *Langzeitbeobachtung von TeV-Blazaren - Qullenanalyse mit MAGIC und Konzeption eines dedizierten Teleskops*, TU Dortmund, Diplomarbeit, 2008 (Zitiert auf Seite 5.)

[6] Ben-Kiki, O. ; Evans, C. ; Ingerson, B.: *YAML Ain't Markup Language (YAML™) Version 1.1*. http://yaml.org/spec/. Version: 28.12.2004 (Zitiert auf Seite 31.)

[7] Breiman, L.: Machine Learning 45(1), 5. (2001) (Zitiert auf Seite 12.)

[8] Bretz, T.: *Observations of the Active Galactic Nucleus 1ES 1218+304 with the MAGIC-telescope*, Julius-Maximilians-Universität Würzburg, Diss., 2006 (Zitiert auf den Seiten 1, 5, 6, 12 und 29.)

[9] Eiben, A.E. ; Smith, J.E.: *Introduction to Evolutionary Computing*. 2. Springer, 2007 (Natural Computing Series). – ISBN 978–3–540–40184–1 (Zitiert auf den Seiten 20 und 53.)

[10] Fix, E. ; Hodges, J.: Discriminatory analysis - nonparametric discrimination: Consistency properties / U.S. Airforce, School of Aviation Medicine. Randolph Field, TX, 1951 (21-49-004.4). – Forschungsbericht (Zitiert auf Seite 17.)

[11] Gaug, M.: *Calibration of the MAGIC Telescope and Observation of Gamma Ray Bursts*, Universitat Autonoma de Barcelona, Diss., 2006 (Zitiert auf den Seiten 7, 10 und 51.)

[12] Hastie, T. ; Tibshirani, R. ; Friedman, J.: *The Elements of Statistical Learning*. Springer, 2009 (Zitiert auf den Seiten 14, 15, 16, 17 und 19.)

[13] Heck, D.: Das Luftschauer-Simulationsprogramm CORSIKA und hadronische Wechselwirkungsmodelle. In: *FZKA-Nachrichten 33* (2001) (Zitiert auf Seite 13.)

[14] Heck, D. ; Pierong, T.: *Extensive Air Shower Simulation with CORSIKA: A User's Guide, Version 6.9xx*. Karlsruhe: Forschungszentrum Karlsruhe GmbH, 2009 (Zitiert auf Seite 12.)

[15] Hillas, A. M.: Cerenkov Light Images of EAS Produced by Primary Gamma Rays and by Nuclei. In: *19th International Cosmic Ray Conference ICRC*, Jones, F.C., 1985 (Zitiert auf den Seiten 1, 9 und 41.)

[16] Li, T. ; Ma, Y.: Analysis methods for results in gamma-ray astronomy. In: *ApJ* (1983) (Zitiert auf Seite 39.)

[17] Liu, H. ; Liu, H. L. (Hrsg.) ; Motoda, H. (Hrsg.): *Feature Extraction, Construction and Selection: A Data Mining Perspective*. 1. Berlin : Springer, 1998 (Kluwer International Series in Engineering & Computer Science). – ISBN 0792381963 (Zitiert auf Seite 19.)

[18] Mierswa, I.: Evolutionary Learning with Kernels: A Generic Solution for Large Margin Problems. In: *Proc. of the Genetic and Evolutionary Computation Conference (GECCO 2006)* (2006) (Zitiert auf Seite 16.)

[19] Mierswa, I. ; Wurst, M. ; Klinkenberg, R. ; Scholz, M. ; Euler, T.: YALE: Rapid Prototyping for Complex Data Mining Tasks. In: Ungar, L. (Hrsg.) ; Craven, M. (Hrsg.) ; Gunopulos, D. (Hrsg.) ; Eliassi-Rad, T. (Hrsg.): *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA : ACM, August 2006. – ISBN 1–59593–339–5, 935–940 (Zitiert auf den Seiten 1 und 13.)

[20] Ritthoff, O. ; Klinkenberg, R. ; Fischer, S. ; Mierswa, I.: A Hybrid Approach to Feature Selection and Generation Using an Evolutionary Algorithm / Collaborative Research Center 531, University of Dortmund. Dortmund, Germany, 2002 (CI-127/02). – Forschungsbericht. – ISSN ISSN 1433–3325 (Zitiert auf den Seiten 21, 22, 23 und 53.)

[21] ROOT Team, The: *ROOT User's Guide*. Geneva: CERN, 2009 (Zitiert auf den Seiten 1, 25 und 26.)

[22] Vapnik, V.: *The Nature of Statistical Learning Theory*. New York : Springer, 1996 (Zitiert auf Seite 13.)